Math

# A GPSS SIMULATION OF THE 360/75 UNDER HASP AND O.S. 360

by

Fred Salz

June 1972

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

# A GPSS Simulation of the 360/75 under HASP and O.S. 360

by

Fred Salz

Department of Computer Science
University of Illinois
Urbana, Illinois

(This research was supported in part by the National Science Foundation under Grant No. NSF GJ 28289).

I would like to thank Bob Skinner for his assistance in supplying the necessary information regarding the I.B.M. 360/75.

## ABSTRACT

The success or failure of a computing system in today's highly competitive market is often determined by the efficiency of its operating system. Consequently, existing operating systems are constantly being modified, extended, and, hopefully, improved. The key question pertaining to the implementation of proposed changes is "Does the proposed change improve the existing operating system?" One appealing method of answering this question is to simulate the operation of the computing system both under the existing operating system and the system with the proposed changes included. The obvious first step in such a study is to build a model to simulate, with accuracy, the existing system. In this paper, such a model is presented for the IBM 360/75 operating under HASP and O.S. A GPSS simulation of the system is presented and some results are given which verify the accuracy of the simulation.

## I. Introduction

In order to study the effect of new computer system proposals, a number of techniques may be employed. One method is to write the new feature into the existing system and run tests. Obviously, for a complex change, this would be extremely costly in both the programmer's time, and unproductive machine time. An analytical model could be devised based on queuing theory analyses, but it would become extremely burdensome for a complex situation. A third method, and the one to which this paper is directed, is to simulate the proposed changes on the computer system currently in use. This results in a minimum of time that is not spent running users programs during the analysis.

For a simulation to be of value however, it must be accurate both statistically and functionally. In order to be certain that analysis of changes based on the simulator are realistic, the model performance must be measured against a known quantity, i.e., the existing system.

For this reason, the model described in this paper is a representation of the IBM 360/75 operating under HASP and O.S. Once it has been shown that statistics from the two sources agree, the model can be used for its intended purpose, that is, an evaluation of new proposals.

## II. Model Description

### A. System operation

#### 1. HASP Initiation

Jobs are fed into the system simultaneously from terminals, tape, readers, disks, and other devices. As a job arrives, it is placed onto the HASP SPOOL (which has a limit of 400 jobs). If the spool is full, either input unit is detached, or the job is recycled back out to tape to be re-read later at a controlled rate.

Once spooled, the job is given a HASP class. In the case of the U. of I. system, the class is assigned based on estimates of CPU time, I/O requests, and lines printed. Each job must go through a sequence of events in a set order, i.e. initiation, execution, printed output, termination, etc. This sequence is controlled by a progression list, which is included in the job information data. The job waits on the SPOOL until selected by a HASP initiator

Each of the 7 initiators can be set to recognize up to 4 different classes of jobs, in a specific order. It is in this order, that a free initiator will take a job off the SPOOL and feed it to O.S. For example, if an initiator is set CBA, it will first search the spool for a class C job, if not found it will look for a class B. If there is no B job, and no A job either, the initiator will be put in a wait state.

Once the job is selected, it is put on the O.S. QUEUE to be serviced by the operating system.

#### 2. O.S. Initiation

Once a job is placed on the O.S. queue, there is no longer any class distinction. There is another set of initiators that select jobs in a first-come first-served manner and removes them from the O.S. queue. It is the function of these initiators to take the job through the various stages of execution.

The JCL for the first (or next) step is scanned for errors, and if everything is satisfactory, data management is called to allocate devices as described on the DD statements. The initiator waits for completion.

The O.S. Supervisor is then called to allocate core space. The first block of contiguous core large enough to contain the step request is allocated to the job. If no such space is available, the initiator must wait, and is therefore tying up both the OS and HASP initiators. No procedures exist for compacting core to avoid fragmentation.

Once core is allocated, the program is loaded, and the job is placed on a ready queue with the highest nonsystem priority.

3. O.S. Scheduler

Jobs are selectively given control of the CPU by the O.S. Scheduler. The job with the highest dispatching priority is given control until an interrupt occurs - either user initiated or system initiated.

HASP - Dispatcher

Every two seconds,a signal is sent by the dispatcher which interrupts the CPU if busy. All of the jobs on the ready queue are then reordered by the assignment of new dispatching priorities based on utilization in the previous 2 second interval. The job that has the lowest ratio of CPU time to I/O requests will get the highest dispatching priority. e.g. - the jobs that used the least CPU time will tend to get the CPU first on return from the interrupt.

During this time, HASP updates elapsed statistics, and checks these against job estimates, and will terminate the job if any have been exceeded.

4. HASP - Termination

When execution of the job is completed, control is returned to the HASP initiator to proceed with job termination. Accounting is updated, the progression list is set to completion, and Print or Punch service is called to produce the actual output. Purge service is then called to physically remove the job from the system.

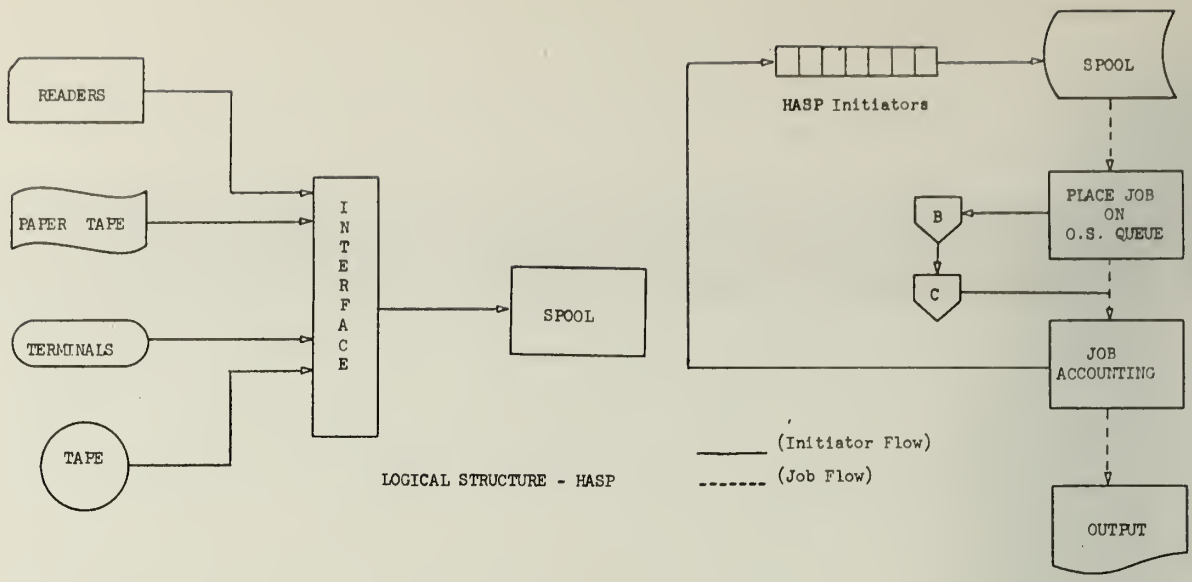
The initiator is then returned to a free state to select a new job from the spool.

5. Express

Since express is available on the 360/75, it had to be represented in the simulation. There are two major differences between events as a job goes through express rather than HASP. First, an express job does not request core, since a piece is dedicated to it at all times. Therefore, in the simulation model, when an express job is initiated, the core allocation routine is bypassed.

Since devices are not allocated, and other system procedures are not necessary, the overhead time normally associated with these functions is not executed. This will be fully discussed in the next section.

Figure 1 shows the system as modelled.




LOGICAL STRUCTURE - HASP

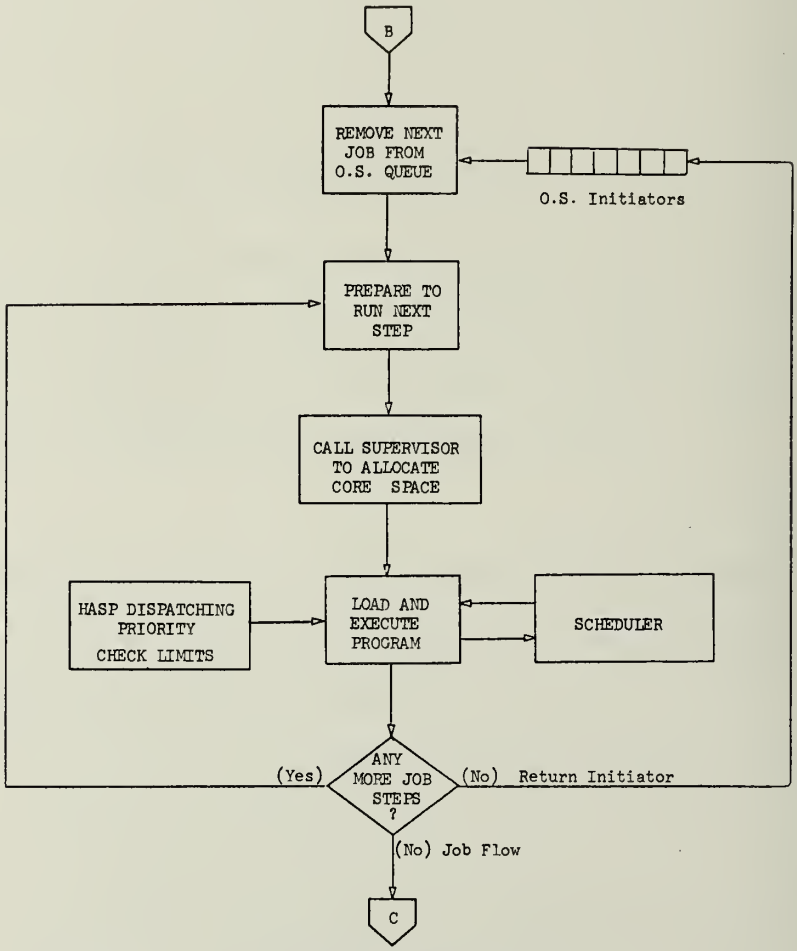



Figure 1 LOGICAL STRUCTURE - O.S. 360

B. General Model Theory

1. Representation of O.S. entities by GPSS.

Before constructing the model of the 360/75, it was necessary to decide how specific entities of the real system would be represented by the model. The method used had to both resemble the real system in its operation, and be compatible with the rest of the GPSS program.

The first issue that had to be decided was the length of the clock unit. Since only one clock interval length could be used throughout the simulation, a unit that was small enough to represent very small time slices while allowing larger units of time (such as 1 hour) had to be chosen. It was decided that each GPSS clock unit would be one millisecond. Since there are events that require less than 1 ms. of time to complete, and this time had to be accounted for, it was included in the general system overhead.

For each job run, a certain amount of time is spent in initiation/ termination, reordering of dispatching priorities, etc. This could have been handled in one of two ways, the first being a strict addition of one or two milliseconds each time a certain operation by the system was to be performed. The second is to account for all system time in one piece, at some point in execution. The later was chosen for the following reasons:

It is not accurately known how much time is spent performing each task (e.g., one millisecond or two, etc.). Therefore, each time this time is used, an error will be added. For example, in five minutes of simulated time, approximately 11,000 time slices would be run, the chains would be reordered 150 times, etc., with each function involving many segments in which overhead must be acccunted. If there is a 10% error per operation amounting to 1 millisecond, with say 10 of these operations per time slice, this would result in at least a 110 second error in this area alone, or 3.6 seconds per job, (based on 30 jobs run in this time).

On the other hand, if the same error occurs in the IBM statistics showing an average of 5 seconds initiation/termination time and system overhead this would result in only a 500 millisecond per job error. In the same period of simulated time, 30 jobs will complete execution, resulting in a total error of only 15 seconds.

Therefore, I have considered that each job (with express as an exception) must execute a fixed amount of overhead. This is accomplished at the end of execution of all job steps. This time is then broken into two pieces - one part being executed while the job retains its current core allocation, and one part executed after deallocation has occurred.

Since the dispatcher on the other hand only performs its task once every 2 seconds, a constant overhead time for that section is defined and executed.

It should be noted that all overhead times, as other constants, are initialized as variables and can be changed as more accurate information becomes available. The logical structure of the simulator is shown in Figures 2a and 2b.

1. Jobs

A job is represented by one GPSS transaction with 48 parameters. The parameters, shown in Table 1, are referenced throughout the simulation to keep a record of what was done, and indicate what the next step will be. In this way, by moving the "job" from one section of the model to another, it could indicate different stages of execution.

2. HASP and O.S. Initiators

In reality, there are two sets of initiators, one for HASP, and another for O.S. They each require the same information about the jobs they are servicing,

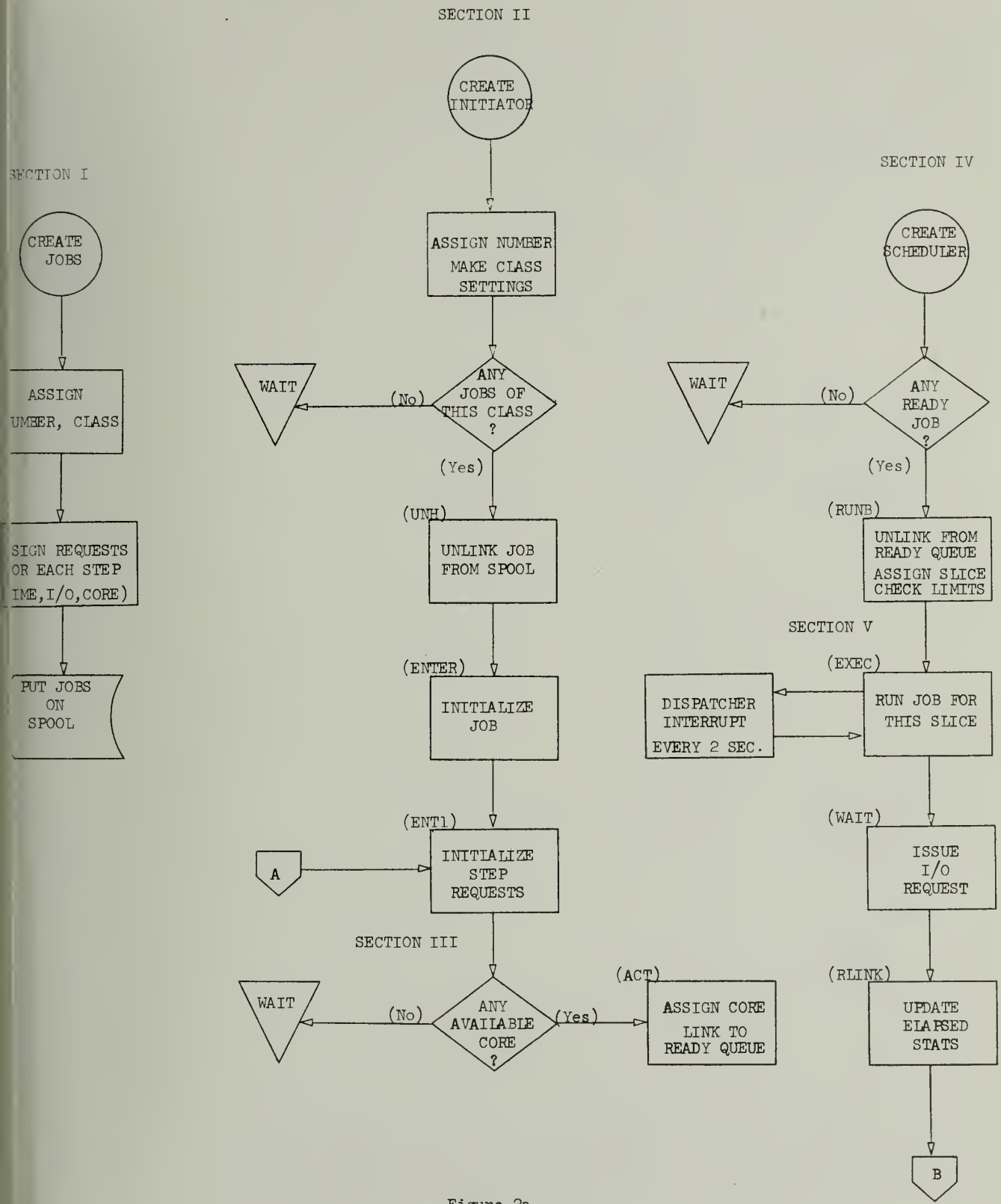
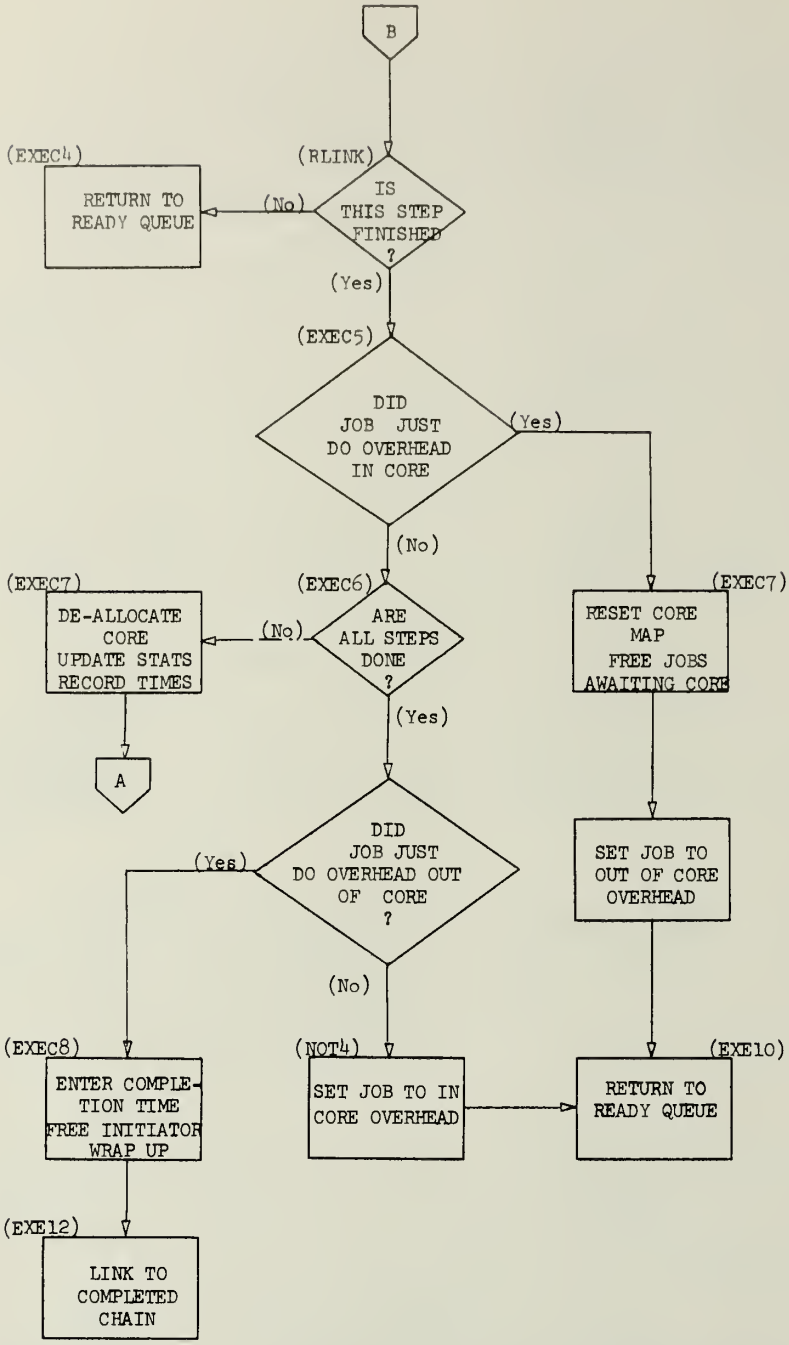

SECTION I
CREATE JOBS
ASSIGN UMBER, CLASS
SIGN REQUESTS OR EACH STEP IME,I/O,CORE)
PUT JOBS ON SPOOL
SECTION II
CREATE INITIATOR
ASSIGN NUMBER MAKE CLASS SETTINGS
WAIT
(No)
ANY JOBS OF THIS CLASS ?
(Yes)
(UNH)
UNLINK JOB FROM SPOOL
(ENTER)
INITIALIZE JOB
(ENT1)
A
INITIALIZE STEP REQUESTS
SECTION III
WAIT
(No)
ANY AVAILABLE CORE ?
(Yes)
(ACT)
ASSIGN CORE LINK TO READY QUEUE
SECTION IV
CREATE SCHEDULER
WAIT
(No)
ANY READY JOB ?
(Yes)
(RUNB)
UNLINK FROM READY QUEUE ASSIGN SLICE CHECK LIMITS
SECTION V
(EXEC)
DISPATCHER INTERRUPT EVERY 2 SEC.
RUN JOB FOR THIS SLICE
(WAIT)
ISSUE I/O REQUEST
(RLINK)
UPDATE ELAPSED STATS
B

Figure 2a




SECTION VI

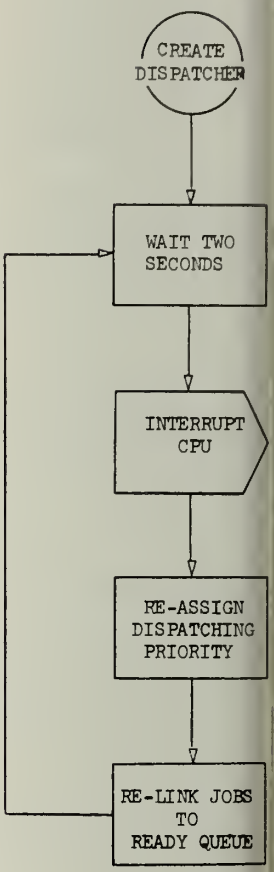



Figure 2b

and the HASP initiator for a specific job must be dormant while the O.S. initiator is running and vice versa. Therefore, 7 transactions were created, each of which represents both HASP and O.S. initiator.

Upon creation, each initiator is assigned up to 4 class settings, which could be changed at any time. Class A=1, ..., express=5. They are then put in an inactive state awaiting the arrival of jobs into the system. After the initiator completes initiation of a job and places it on the ready queue, the HASP initiator becomes an O.S. initiator.

It is the O.S. initiator that flows through the core allocation routine to request core space, and finally places the job on the ready queue to run. This initiator then becomes dormant waiting for the job (or job step) to complete.

When the entire job completes, this initiator is returned to the section of the simulation where it again performs the function of HASP.

3. Queues

All HASP and O.S. queues are represented by user chains. This allows ordering of the objects on the queues, while gathering waiting time and size statistics, and allows arbitrary manipulation of the elements on the chains.

There is a separate chain for each of the five classes of jobs (1-5). This allows for an efficient method of checking if any jobs of the desired class exists, by merely checking the length of that chain. This scheme also returns waiting statistics broken down into each of the four classes, since each chain is independent.

Whenever an initiator or job is to be put in a wait state, it must be taken off the current events chain, and therefore, it is placed on a chain

representing that condition. Initiators waiting for jobs are put on chain 6, and initiators waiting to assign core go to chain 9. When a job arrives, all transactions are unlinked from 6, and when any job deallocates core, chain 9 is emptied.

Jobs that are waiting for core are placed on chain 5. When its initiator is notified that core has been allocated, the waiting job is put on the ready queue (chain 8) according to the contents of Parameter 7 (dispatching priority) to execute. Representing this queue as a chain, allows for this ordering.

4. CPU Usage - Scheduler

The scheduler has the responsibility of determining which task will next get control of the CPU. The scheduler is represented by one high priority transaction that unlinks jobs from the chain they are waiting on, and lets them seize facility 1 thorugh 4 depending on the current status of the job. While advancing the clock in the facility, no other transactions are permitted to do so. Other transactions may, however, enter advance blocks representing I/O requests, and other system processes may take place. This is a rational approach since many functions in the real system are actually carried out in parallel.

When a transaction releases the facility, control is returned to the scheduler which gives the next ready job control of the CPU.

5. The Dispatcher

The HASP dispatching priority reassignment is carried out by one final transaction. Every "2 seconds", this transaction is given control of the simulator, and proceeds to reorder the jobs on the ready queue (chain 8) and the jobs waiting for I/O requests (chain 11). The job currently in control of the CPU (if any) is interrupted, and placed back on the ready queue according to its new priority.

When all of the reordering is complete, the scheduler is freed, and the dispatcher is made dormant for another two seconds.

Communication between Program Sections

For any of a number of reasons, certain sections of the simulator must operate in synchronization with other sections. The most frequent example of this situation occurs when a transaction that is not representing a job (i.e., an initiator, dispatcher, etc.) must access information stored in a job transaction's parameter. Since no transaction can access the parameters of another, the "job" must be given control of the simulator, while the transaction desiring the information is forced to wait. In this situation, the transaction desiring the information sets on logic switch, and waits for it to be reset, or it enters a buffer block,depending on the relative priorities of the transactions. Since the transaction containing the information is now on the current events chain, it is given control of the simulator. It then passes the desired information to the requesting transaction, usually through a savevalue or matrix savevalue, or it changes information within itself (such as dispatching priority). When completed, the logic switch is reset, and control is given back to the requesting transaction. Examples of this type of communication can be seen following block UNH in section II, as information about a job just selected is passed back to the initiator.

Another area that requires synchronization is the scheduler. Once a job is given control of the CPU, no other one may do so until the previous one has completed. Again, a logic switch is set by the scheduler(#51) when a job is given the CPU, and the job resets it when completed.

More discussion in detail will be found in later sections.

C. Specific Model Operation

In order to fully understand the detailed operation of the simulator the following paragraphs describe the individual sections of the program, and should be used in conjunction with the program flow chart and listing in appendices A and B. Specific blocks are referenced by block name. The word in parentheses on the listing at the beginning of each program segment designates the type of transaction flowing in that section.

Section I - Creation of Job Stream.

Each of the two generate blocks in this section create transactions with a priority of 100 (for reasons that will become evident later), and 48 fullword parameters.

To look at the system at the normal load level, without waiting for the queues to build up, initial jobs are created (NOT1) and placed on the appropriate chains. A separate class assignment is made since no express jobs originate on queues.

The generate block at (NOT2) creates transactions exponentially with a mean interarrival rate defined in V60. The transactions from both generate blocks continue from block BEGIN.

As a job is created, the transactions that are waiting on chain six (initiators) are unlinked to test the new job. Since the jobs have a priority of 100, and the initiators a priority of 50, all of the enterring jobs will link themselves to chains 1-5 before the initiators gain control of the simulator, which is the desired sequence of events. The assignments that are made at this point are commented on in the listing.

The PRIORITY block at INITF is a dummy block that sets the status change flag. This insures that if the arrival of this job effected any higher priority transactions that already gave up the simulator, they will get another chance to run.

The jobs are then linked onto the chain for their job class, whose number is in parameter 2, and the unlinked initiators are free to run.

Section II - Creation of Initiator/Terminators

Seven initiator/terminators are created with a priority of 50, and containing the default of 12 fullword parameters. The assignments made at NOT3 come from the temporary information stored in matrix INFO, and the cells are cleared when the information is received. The class designations are stored in Parameters 2-5.

At UNH1, the initiator checks the chain corresponding to its primary class for the availability of a job. If the chain is empty the other classes in P3-P5 are checked beginning at TEST1.

Section IIa

If no suitable jobs are found, the initiator is linked to chain 6 to await the arrival of new jobs.

When a match is found, the class of that job is stored in savevalue. TEMP1, and one job from job from that chain is unlinked and sent to ENTER. In order to pass the initiator number to the job, it is stored in savevalue 6.

Control now must be given to the job in order to pass information to the initiator and matrix INFO, as well as to initialize the first job step. This is accomplished by the blocking process described in its place earlier. (i.e. in this case a BUFFER block).

The job passing through section IIb makes the assignments described on the listing. The blocks from ENTER to ENT1 are executed only at job initialization. Those following ENT1 make the initializations for each step, and therefore are executed at the beginning of each step.

Parameter 12 is used frequently in the model to allow indirect addressing of other parameters. The intent of the block at ENT1 and the one following is to access information stored in the parameter whose number is determined by V41. The method employed here is the most efficient allowed by GPSS. The job is then linked on a chain waiting for core to be assigned.

When control is returned to the initiator, it receives the information passed to it, and transfers to section III to request core. (UNH2)

Section III - Core Allocation

Since the system essentially contains 600k of core allocated in a minimum of 2k chunks, core in the model is represented as 300 units. If EXPRESS is simulated, this figure is reduced to 200. The algorithm used is that the FIRST available CONTIGUOUS block large enough to contain the job is allocated. The logic behind blocks REQN through REQF can best be seen on the flow chart. The four blocks surrounding REQA loop through the section of "core" allocated to the job, placing the job number in the cell. This prevents the cell from being allocated to any other job, and can give a pictoral representation of the core map if desired.

The time the request was filled is passed to the job, and the job is released from chain 5 and sent to section IIIa to be prepared for running, while the initiator links itself to chain 7 to await completion of execution.

Section IIIa determines the mean running time between I/O requests, and records other information about its current status. The job then signals the scheduler that there is something on the ready queue, (resetting logic switch 50) and links itself to chain 8 according to dispatching priority.

Section IV - Control Section

The control section of the program insures that only one task uses the CPU at any one time. All transactions that use the CPU (except the dispatcher) are on chain 8, and only section IV unlinks transactions and sends them to section V to execute.

The ready queue (chain 8) is tested for the presence of a job. If none is found, logic switch 50 is set to block the scheduler, until another job enterring the queue resets it. If a job is present, it is sent to section V (EXEC) to run, and the scheduler is blocked until the job interrupts.

When the job returns control, the scheduler is free to look for the next job, even though the previous one has not returned to the queue. This is further discussed in the details of section V.

Section V - Job Execution

Jobs that are to be given control of the CPU are sent to EXEC, where preliminary actions are taken before running. Savevalue SLICE is assigned the number of milliseconds the job will run before interruption due to an I/O request. This is determined by an exponential distribution about the mean in parameter 17,assigned in section IIIa. The minimum slice allocated is one millisecond.

The block at EXECA tests to see if the elapsed time for the step plus the new slice is greater than the total time requested for the step.

If so, the slice is reduced to the remaining time until completion. If not, the facility whose number is in P48 (P48 is the job status indicator) is seized. The facility entity was used to represent the CPU for a number of reasons. First, the statistics gathered by a facility are the most valuable in this case (Percent utilization, etc.). Secondly, the facility may be interrupted, thereby allowing the dispatcher to reorder the ready queue every 2 seconds. (See section VI). And thirdly, by using different facilities for different functions (specified in P48) separate statistics can be accumulated.

Once the facility is seized, the slice is advanced, the elapsed times are updated, and control is returned to the scheduler by resetting logic switch 51. Parameter 45 and 46 contain the number of milliseconds and I/O requests respectively issued since the last dispatcher interrupt. Note that even though the scheduler is freed to give another job the CPU, the current job is not returned to the ready queue.

Blocks EXEC3 through EXEC4 simulate the I/O request time. In order to facilitate the reassignment of dispatching priorities, each transaction issuing an I/O request is split in two. Since jobs may be in an advance block when the interruption occurs, they cannot be removed to do the reassignment. Therefore, the original transaction is placed on chain 11 where the dispatcher is free to unlink it and link it back, while the copy waits in the advance block. When the time for the I/O request is complete, the copy unlinks the original, sends it to RLINK, and terminates. At RLINK, the step wait time is updated and if the step hasn't completed it is relinked to the ready queue.

To understand the logic of the procedures followed when a step completes, it is necessary to refer to the general flow diagram in Appendix B. The actions taken under different conditions is described below, and may be taken in an order not identical to the block diagram.

a. Job just completed execution of step, but all steps not complete. (EXEC 7). If the number of elapsed I/O requests is less than the number requested, the remaining requests are issued. The core space given to this step is deallocated and all jobs waiting for core are free to inspect this new segment. The core allocation information in INFO is cleared, and the job is sent to EXEC9, where step statistics are recorded. The job is now transfered to ENT1 in section IIb for the new step requests to be initialized.

b. Job has just finished executing all of its steps (NOT 4). Appropriate indicators (in INFO and P48) are set to indicate the job is to run its overhead while in core (see section J). The step end stats are seconded, initializations are made so the overhead can be run, and the job is placed back on the ready queue.

c. Job has just finished running overhead in core. (NOT 5). Indicators are set for running overhead out of core, and initializations are made to run overhead out of core. The job is placed back on the ready queue, after core is deallocated.

d. Job has just finished overhead out of core (EXEC 8). This marks the end of execution for this job, and all final statistics are recorded, as commented on the program listing. The initiator for this job is freed, and the job is placed on chain 10 to be printed out at the end of the run.

Section VI - The Dispatcher

f. The dispatching priority reassignment as previously described is carried out by one final transaction flowing through the "dispatching section". When the transaction awakes after two seconds, it interrupts the CPU.

The Facility representation was chosen for the CPU so that it could be preempted by the dispatcher. The preemption merely removes the job in the advance block and places it back on the ready queue, without allowing the scheduler to continue running jobs. In this way, all jobs not issuing I/O requests are in the same place (i.e. on the ready queue) so their priorities can be revised.

All of the jobs on the ready queue are unlinked and proceed to change their own priorities. This is the most practical way of accomplishing this, since only the transaction itself can reference its parameters, which in this case contain all of the necessary data.

Once the reassignment is completed, the jobs are relinked onto the ready queue (chain 8), the scheduler is freed, and the dispatcher goes to sleep for another two sections.

D. Discussion of output data

As an aid to understanding some of the statistics available from the simulation, some sample output data is shown in Figures 3-5. The results represent 1 hour of simulated time and required 9 minutes of 360/75 processor time for execution. The amount of core needed to run the simulation will vary depending on what information is kept throughout the run. For example, if all completed job transactions are kept, the larger (256K) version of GPSS will be required, with a certain amount of reallocation (see GPSS operation's manual). If these transactions are terminated, the 116K version would be sufficient, with minor reallocation necessary.

In studying the statistical printout, it should be noted that the arrival rates used are derived from a monthly average, and therefore utilization figures, queue sizes, etc., will seem low for a normal daytime period. For details of times and constants used, see the initializations and variable definitions on the program listing in appendix B.

The facility statistics are shown in Figure 3. Facility 1 represents the CPU time used for user problem programs, and shows a 19.6% utilization. The number of entries (63, 121) are the number of time slices allocated, with the average being 11.224 milliseconds.

| FACILITY | AVERAGE UTILIZATION | NUMBER ENTRIES | AVERAGE TIME/TRAN | SEIZING TRANS. NO. | PREEMPTING TRANS. NO. |
|---|---|---|---|---|---|
| 1 | .196 | 63121 | 11.224 | | |
| 2 | .082 | 551 | 541.948 | 241 | |
| 3 | .082 | 534 | 557.115 | | |
| 4 | .024 | 1757 | 50.000 | | |
| 5 | .362 | 63558 | 20.525 | 241 | |

Figure 3

Facilities 2 and 3 represent the CPU usage for in core and out of core overhead discussed earlier, and Facility 4 is the dispatcher's use of the CPU. The total of 1, 2, and 3 is shown in Facility 5.

User chains 1-5, shown in Figure 4, are the queues for job classes A-EXPRESS as discussed in section II-B-d. The average time/transaction entry is the expected waiting time, and the average contents is the expected number on each of the queues. Note that there was no class 4 initiator set, and therefore, the current and maximum contents for class 4 are equal.

| USER CHAIN | TOTAL ENTRIES | AVERAGE TIME/TRANS | CURRENT CONTENTS | AVERAGE CONTENTS | MAXIMUM CONTENTS |
|---|---|---|---|---|---|
| 1 | 54 | 21907.886 | | .328 | 8 |
| 2 | 59 | 291859.437 | | 4.783 | 11 |
| 3 | 10 | 163491.000 | 1 | .454 | 2 |
| 4 | 17 | 1972960.000 | 17 | 9.316 | 17 |
| 5 | 104 | 1890.451 | | .054 | 2 |
| 6 | 659 | 15618.820 | 4 | 2.859 | 6 |
| 7 | 258 | 41080.484 | 3 | 2.944 | 6 |
| 8 | 65457 | 58.274 | 2 | 1.059 | 6 |
| 9 | 187 | 23039.699 | | 1.196 | 4 |
| 10 | 223 | 931415.375 | 105 | 57.695 | 118 |
| 11 | 56606 | 96.804 | | 1.522 | 6 |
| 13 | 154 | 27976.777 | | 1.196 | 4 |
| 14 | 118 | 1800000.000 | 118 | 59.000 | 118 |

Figure 4

Chain 8 is the ready queue and shows that the average job had to wait 58.274 milliseconds before regaining control of the CPU after an I/O request. For details of the other chains, see Table 1.

Tables 1-5, Figure 5 , measure the average turnaround time for each job class. The mean argument is the expected time in the system.

Figure 6 shows a few sample completed jobs. With the aid of Table 2 , the information available is evident.

TABLE 1

| ENTRIES IN TABLE | MEAN ARGUMENT | STANDARD DEVIATION | SUM OF ARGUMENTS | |
|---|---|---|---|---|
| 54 | 86475.750 | 79872.000 | 4669692.000 | NON-WEIGHTED |

| UPPER LIMIT | OBSERVED FREQUENCY | PER CENT OF TOTAL | CUMULATIVE PERCENTAGE | CUMULATIVE REMAINDER | MULTIPLE OF MEAN | DEVIATION FROM MEAN |
|---|---|---|---|---|---|---|
| 0 | 0 | .00 | .0 | 100.0 | -.000 | -1.082 |
| 25000 | 16 | 29.62 | 29.6 | 70.3 | .289 | -.769 |
| 50000 | 9 | 16.66 | 46.2 | 53.7 | .578 | -.456 |
| 75000 | 5 | 9.25 | 55.5 | 44.4 | .867 | -.143 |
| 100000 | 5 | 9.25 | 64.8 | 35.1 | 1.156 | .169 |
| 125000 | 3 | 5.55 | 70.3 | 29.6 | 1.445 | .482 |
| 150000 | 4 | 7.40 | 77.7 | 22.2 | 1.734 | .795 |
| 175000 | 3 | 5.55 | 83.3 | 16.6 | 2.023 | 1.108 |
| 200000 | 4 | 7.40 | 90.7 | 9.2 | 2.312 | 1.421 |
| 225000 | 1 | 1.85 | 92.5 | 7.4 | 2.601 | 1.734 |
| 250000 | 0 | .00 | 92.5 | 7.4 | 2.890 | 2.047 |
| 275000 | 3 | 5.55 | 98.1 | 1.8 | 3.180 | 2.360 |
| 300000 | 0 | .00 | 98.1 | 1.8 | 3.469 | 2.673 |
| 325000 | 1 | 1.85 | 100.0 | .0 | 3.758 | 2.986 |

REMAINING FREQUENCIES ARE ALL ZERO

TABLE 2

| ENTRIES IN TABLE | MEAN ARGUMENT | STANDARD DEVIATION | SUM OF ARGUMENTS | |
|---|---|---|---|---|
| 57 | 448727.812 | 166400.000 | 25577488.000 | NON-WEIGHTED |

| UPPER LIMIT | OBSERVED FREQUENCY | PER CENT OF TOTAL | CUMULATIVE PERCENTAGE | CUMULATIVE REMAINDER | MULTIPLE OF MEAN | DEVIATION FROM MEAN |
|---|---|---|---|---|---|---|
| 0 | 0 | .00 | .0 | 100.0 | -.000 | -2.696 |
| 25000 | 0 | .00 | .0 | 100.0 | .055 | -2.546 |
| 50000 | 0 | .00 | .0 | 100.0 | .111 | -2.396 |
| 75000 | 0 | .00 | .0 | 100.0 | .167 | -2.245 |
| 100000 | 1 | 1.75 | 1.7 | 98.2 | .222 | -2.095 |
| 125000 | 1 | 1.75 | 3.5 | 96.4 | .278 | -1.945 |
| 150000 | 0 | .00 | 3.5 | 96.4 | .334 | -1.795 |
| 175000 | 1 | 1.75 | 5.2 | 94.7 | .389 | -1.644 |
| 200000 | 2 | 3.50 | 8.7 | 91.2 | .445 | -1.494 |
| 225000 | 0 | .00 | 8.7 | 91.2 | .501 | -1.344 |
| 250000 | 1 | 1.75 | 10.5 | 89.4 | .557 | -1.194 |
| 275000 | 3 | 5.26 | 15.7 | 84.2 | .612 | -1.044 |
| 300000 | 0 | .00 | 15.7 | 84.2 | .668 | -.893 |
| 325000 | 0 | .00 | 15.7 | 84.2 | .724 | -.743 |
| OVERFLOW | 48 | 84.21 | 100.0 | .0 | | |

AVERAGE VALUE OF OVERFLOW 496389.62

TABLE 3

| ENTRIES IN TABLE | MEAN ARGUMENT | STANDARD DEVIATION | SUM OF ARGUMENTS | |
|---|---|---|---|---|
| 8 | 300598.750 | 158464.000 | 2404790.000 | NON-WEIGHTED |

| UPPER LIMIT | OBSERVED FREQUENCY | PER CENT OF TOTAL | CUMULATIVE PERCENTAGE | CUMULATIVE REMAINDER | MULTIPLE OF MEAN | DEVIATION FROM MEAN |
|---|---|---|---|---|---|---|
| 0 | 0 | .00 | .0 | 100.0 | -.000 | -1.896 |
| 40000 | 0 | .00 | .0 | 100.0 | .133 | -1.644 |
| 80000 | 0 | .00 | .0 | 100.0 | .266 | -1.392 |
| 120000 | 2 | 25.00 | 25.0 | 75.0 | .399 | -1.139 |
| 160000 | 1 | 12.50 | 37.5 | 62.5 | .532 | -.887 |
| 200000 | 0 | .00 | 37.5 | 62.5 | .665 | -.634 |
| 240000 | 0 | .00 | 37.5 | 62.5 | .798 | -.382 |
| 280000 | 0 | .00 | 37.5 | 62.5 | .931 | -.129 |
| 320000 | 0 | .00 | 37.5 | 62.5 | 1.064 | .122 |
| 360000 | 1 | 12.50 | 50.0 | 50.0 | 1.197 | .374 |
| 400000 | 2 | 25.00 | 75.0 | 25.0 | 1.330 | .627 |
| 440000 | 0 | .00 | 75.0 | 25.0 | 1.463 | .879 |
| 480000 | 1 | 12.50 | 87.5 | 12.5 | 1.596 | 1.132 |
| 520000 | 1 | 12.50 | 100.0 | .0 | 1.729 | 1.384 |

REMAINING FREQUENCIES ARE ALL ZERO

TABLE 5

| ENTRIES IN TABLE | MEAN ARGUMENT | STANDARD DEVIATION | SUM OF ARGUMENTS | |
|---|---|---|---|---|
| 104 | 10820.679 | 9344.000 | 1125351.000 | NON-WEIGHTED |

| UPPER LIMIT | OBSERVED FREQUENCY | PER CENT OF TOTAL | CUMULATIVE PERCENTAGE | CUMULATIVE REMAINDER | MULTIPLE OF MEAN | DEVIATION FROM MEAN |
|---|---|---|---|---|---|---|
| 0 | 0 | .00 | .0 | 100.0 | -.000 | -1.158 |
| 2500 | 15 | 14.42 | 14.4 | 85.5 | .231 | -.890 |
| 5000 | 18 | 17.30 | 31.7 | 68.2 | .462 | -.622 |
| 7500 | 16 | 15.38 | 47.1 | 52.8 | .693 | -.355 |
| 10000 | 15 | 14.42 | 61.5 | 38.4 | .924 | -.087 |
| 12500 | 6 | 5.76 | 67.3 | 32.6 | 1.155 | .179 |
| 15000 | 9 | 8.65 | 75.9 | 24.0 | 1.386 | .447 |
| 17500 | 3 | 2.88 | 78.8 | 21.1 | 1.617 | .714 |
| 20000 | 6 | 5.76 | 84.6 | 15.3 | 1.848 | .982 |
| 22500 | 3 | 2.88 | 87.4 | 12.5 | 2.079 | 1.249 |
| 25000 | 3 | 2.88 | 90.3 | 9.6 | 2.310 | 1.517 |
| 27500 | 2 | 1.92 | 92.3 | 7.6 | 2.541 | 1.785 |
| 30000 | 2 | 1.92 | 94.2 | 5.7 | 2.772 | 2.052 |
| 32500 | 1 | .96 | 95.1 | 4.8 | 3.003 | 2.320 |
| OVERFLOW | 5 | 4.80 | 100.0 | .0 | | |

AVERAGE VALUE OF OVERFLOW 36629.79

Figure 5

USER CHAIN 10

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 66 | 715798 | 100 | 66 | 569231 | 55 | 1 | 2524 | |
| | | | | | 30 | 0 | 0 | |
| | | | | | 1 | 1 | 0 | |
| | | | | | 668478 | 715798 | 0 | |
| | | | | | 0 | 0 | 3000 | |
| | | | | | 2524 | 30 | 49 | |
| | | | | | 668478 | 695421 | 3000 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 9703 | 0 | 0 | |
| 67 | 735747 | 100 | 67 | 567440 | 54 | 1 | 161 | |
| | | | | | 141 | 0 | 0 | |
| | | | | | 1 | 1 | 0 | |
| | | | | | 658358 | 735747 | 0 | |
| | | | | | 0 | 0 | 12400 | |
| | | | | | 161 | 141 | 51 | |
| | | | | | 658358 | 695421 | 12400 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 22652 | 0 | 0 | |
| 16 | 762756 | 100 | 16 | 1 | 11 | 2 | 8286 | |
| | | | | | 1148 | 0 | 0 | |
| | | | | | 3 | 3 | 0 | |
| | | | | | 453549 | 762756 | 0 | |
| | | | | | 0 | 0 | 113900 | |
| | | | | | 4928 | 154 | 20 | |
| | | | | | 453549 | 453549 | 14500 | |
| | | | | | 2439 | 559 | 69 | |
| | | | | | 493334 | 493334 | 55900 | |
| | | | | | 919 | 435 | 58 | |
| | | | | | 604687 | 665878 | 43500 | |
| | | | | | 20357 | 53014 | 47309 | |
| 76 | 775843 | 100 | 76 | 766206 | 65 | 1 | 3014 | |
| | | | | | 9 | 0 | 0 | |
| | | | | | 1 | 1 | 0 | |
| | | | | | 766206 | 775843 | 0 | |
| | | | | | 0 | 0 | 900 | |
| | | | | | 3014 | 9 | 38 | |
| | | | | | 766206 | 766206 | 900 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 509 | 0 | 0 | |
| 73 | 777691 | 100 | 73 | 764678 | 64 | 5 | 373 | |
| | | | | | 49 | 0 | 0 | |
| | | | | | 1 | 1 | 0 | |
| | | | | | 764678 | 777691 | 0 | |
| | | | | | 0 | 0 | 4900 | |
| | | | | | 373 | 49 | 0 | |
| | | | | | 764678 | 764678 | 4900 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 7740 | 0 | 0 | |
| 79 | 779940 | 100 | 79 | 773737 | 66 | 5 | 1359 | |
| | | | | | 7 | 0 | 0 | |
| | | | | | 1 | 1 | 254 | |
| | | | | | 777691 | 780040 | 0 | |
| | | | | | 0 | 0 | 700 | |
| | | | | | 1359 | 7 | 0 | |
| | | | | | 777691 | 777691 | 700 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 290 | 0 | 0 | |
| 52 | 785046 | 100 | 52 | 625334 | 58 | 3 | 8380 | |
| | | | | | 166 | 0 | 0 | |
| | | | | | 1 | 1 | 0 | |
| | | | | | 697971 | 785046 | 0 | |
| | | | | | 0 | 0 | 16600 | |
| | | | | | 8380 | 166 | 81 | |
| | | | | | 697971 | 733197 | 16600 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 0 | 0 | 0 | |
| | | | | | 21769 | 0 | 0 | |

Figure 6

# PARAMETERS AND INDICES

## HASP/O.S. INITIATORS

Priority· 50
Parameters· 12

1. Initiator Number
2. First Class Preference
3. Second Class Preference
4 Third Class Preference
5. Fourth Class Preference
6.
7.
8. Job number when selected

## USER CHAINS

1. Class 1 Jobs on queue (Jobs)
2. Class 2 Jobs on queue (Jobs)
3. Class 3 Jobs on queue (Jobs)
4. Class 4 Jobs on queue (Jobs)
5. Class 5 Jobs on queue (Jobs)
6. Init wait for Jobs (Init)
7. Init wait for job to run (Init)
8. Ready queue (linked by P7) (Jobs)
9. Wait for Core (Init)
10. Completed Jobs (Jobs)
11. Waiting for I/O requests (Jobs)
13. Waiting for core in O.S. (Jobs)
14. More completed jobs (Jobs)

## MSAVEVALUE INFO (7,10)

Row Number (1-7) is initiator number
Columns· 9

1 Job number
2 Step number
3. Status indicator
4. Core request
5 Base of core allocated
6. Class of job
7 Execution time - this step
8. I/O requests - this step
9. Time of job creation

Job Status Indicator

1. Running requested time
2. Running overhead in core
3. Running overhead out of core
4. Job completed

## MSAVEVALUE CORE (1, X1)

Columns· X1 Savevalue X1 initialized to size of core
1 - X1-1 Job number using this space in core
X1 Dummy space always (-1)

Table 1

# PARAMETERS AND INDICES

## JOBS

Priority 100
Parameters: 48

1. Job number
2. Job class
* 3. Execution time - current step
* 4. Elapsed execution time

* 5. I/O requests - current step
* 6. Elapsed I/O requests
7. Dispatching priority
8. Initiator number when chosen

9. Total Number of steps
10. Current step executing
*11. Work space
* 12. Work space

13. Time removed from spool
14. Time execution completed
15. Number of cards read
16. Number of lines printed

17. Mean time slice
18. Number of I/O req. per slice
* 19. Wait time for I/O requests
20. Turnaround time

21. Execution time - step 1
22. I/O requests - step 1
23. Core request - step 1
24. Core allocation - step 1

25. Step 1 start
26. Step 1 core request filled
27. Wait for I/O this step
28. Step 1 end

29. Execution time - step 2
30. I/O requests - step 2
31. Core request - step 2
32. Core allocation - step 2

33. Step 2 start
34. Core request filled
35. Wait time for I/O requests
36. Step 2 end

37. Execution time - step 3
38. I/O requests - step 3
39. Core request - step 3
40. Core allocated - step 3

41. Step 3 start
42. Core request filled
43. Wait for I/O requests
44. Step 3 end

* 45. Time elapsed since interrupt
* 46. I/O elapsed since interrupt
47. Place of origin of this job
48. Job status indicator

At job completion,

45. Wait time on ready queue - step 1
46. Wait time on ready queue - step 2
47. Wait time on ready queue - step 3

3. Total execution time for Job (CPU)
5. Total I/O request for Job
19. Total wait time for job (I/O)

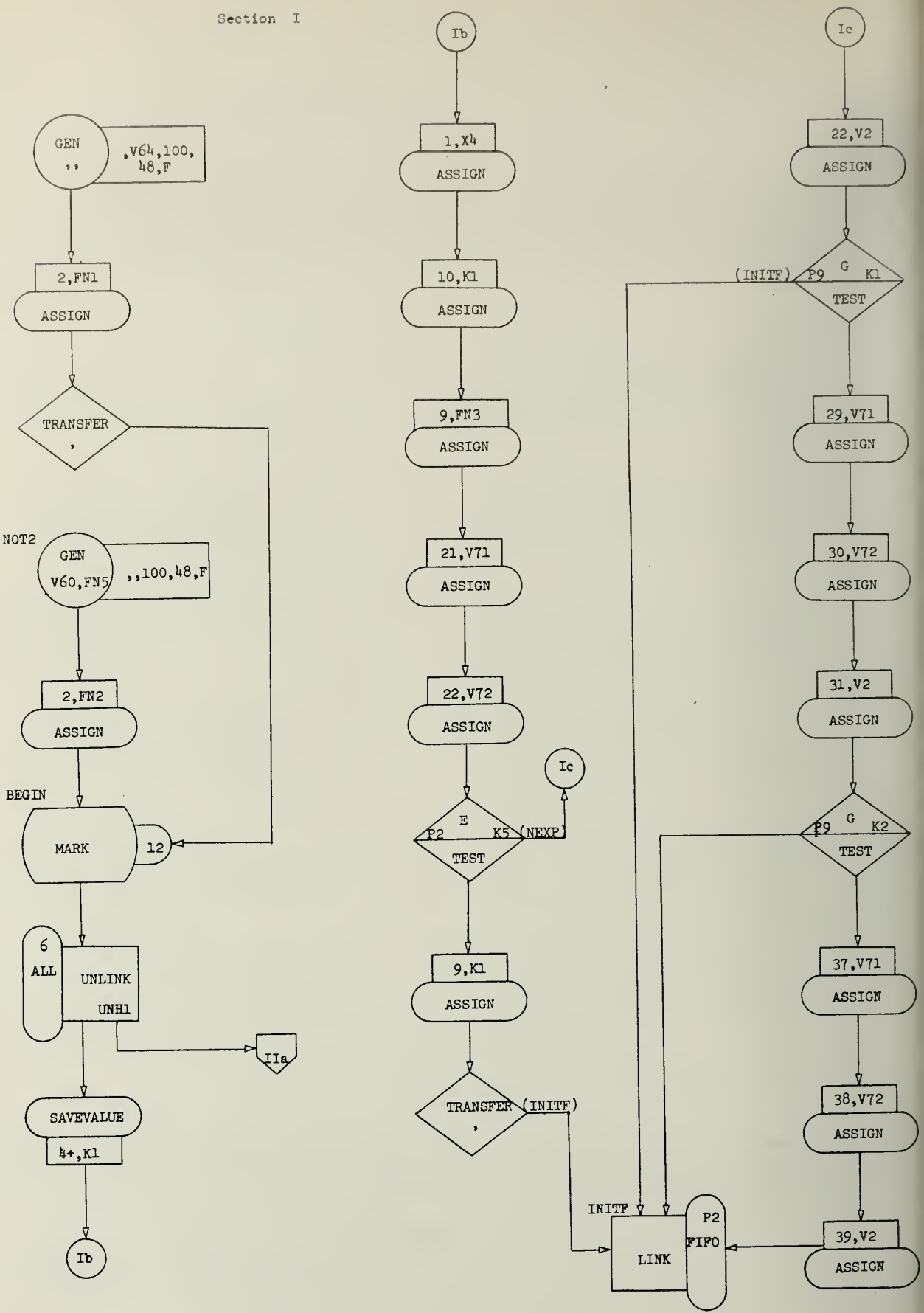
Job Status Indicator

1 - Job executing requested time
2 - Job running overhead in core
3 - Job running overhead out of cor
4 - Job completed

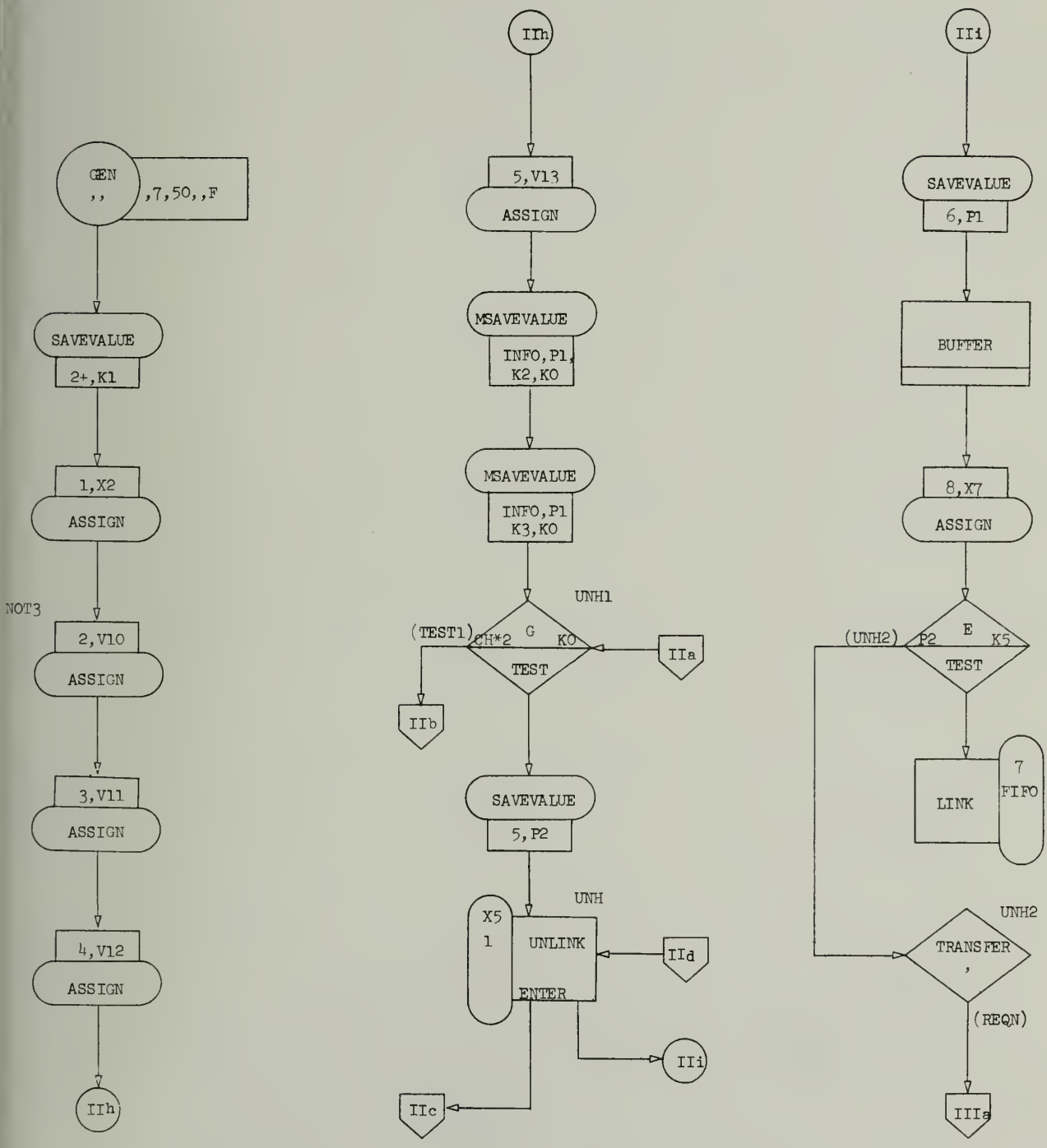
* Parameters indicated by (*) are used during execution for purposes other than what might appear on the final completion chain.

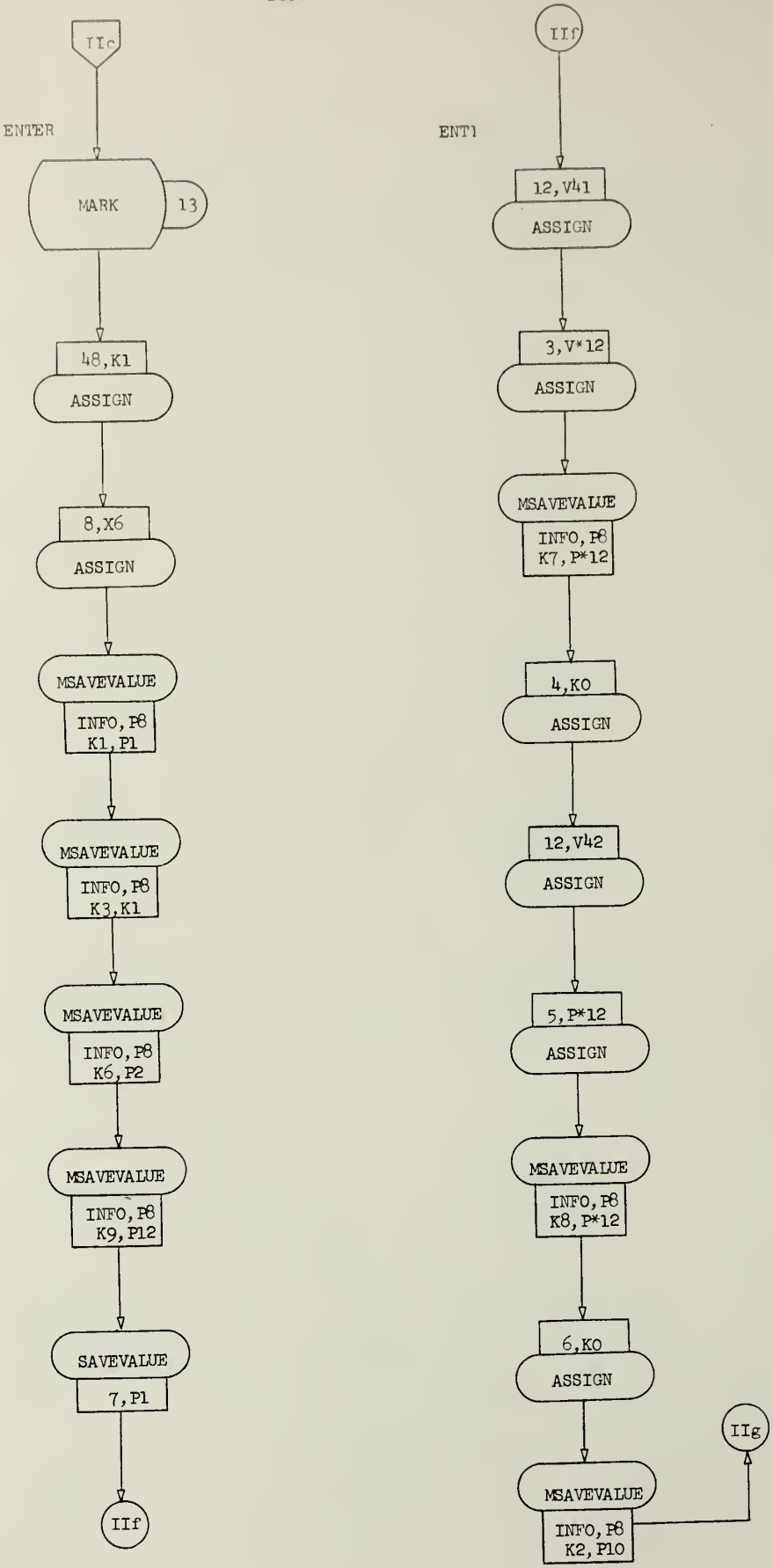
Table 2

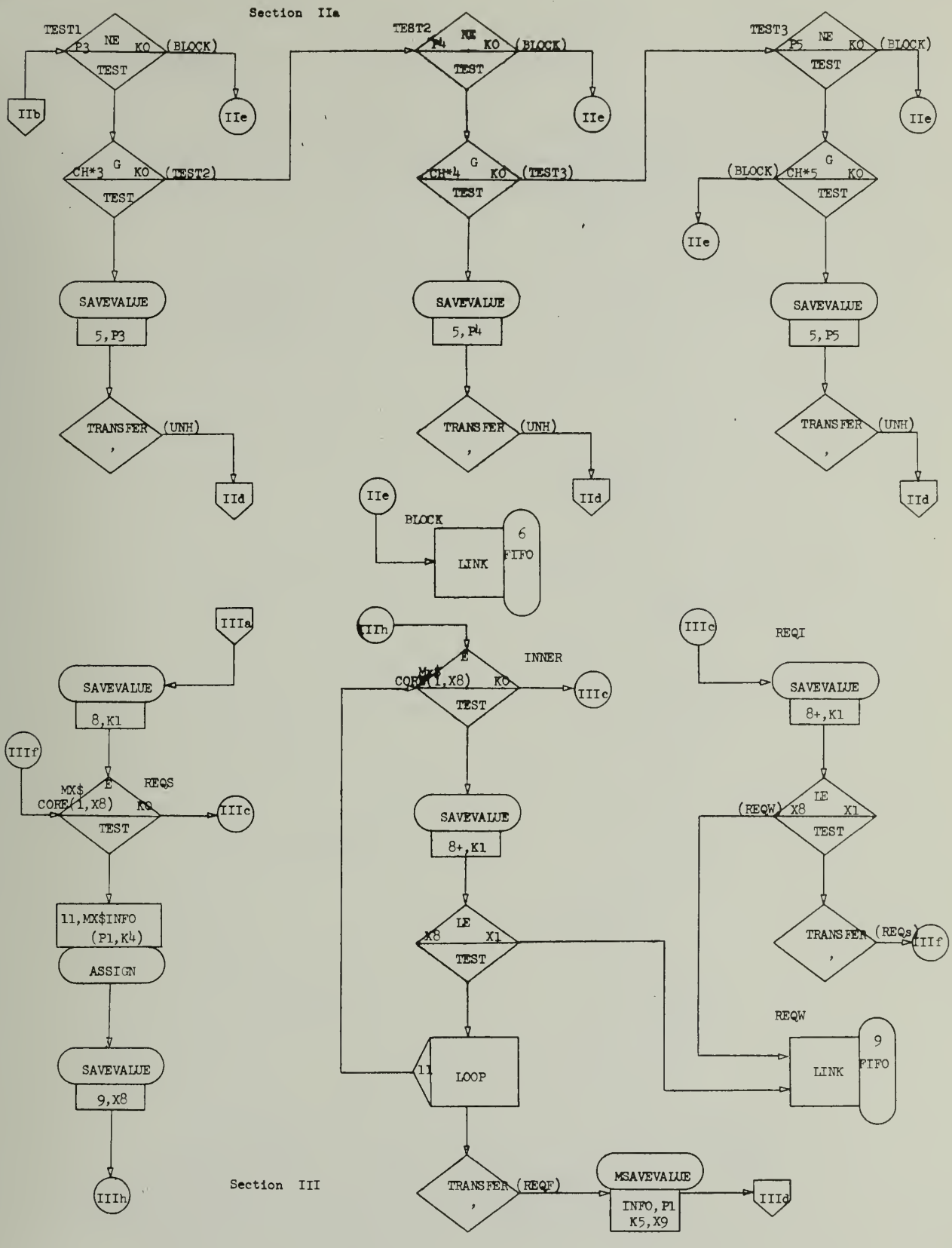
# APPENDIX A

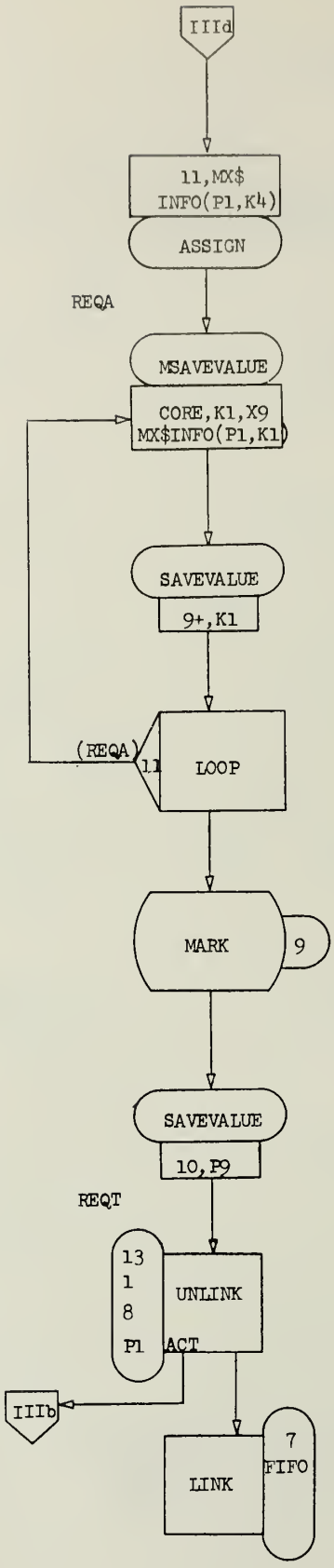

GEN
,,
,V64,100,
48,F
2,FN1
ASSIGN
TRANSFER
,
NOT2
GEN
V60,FN5
,,100,48,F
2,FN2
ASSIGN
BEGIN
MARK
12
6
ALL
UNLINK
UNH1
IIa
SAVEVALUE
4+,K1
Ib
Ib
1,X4
ASSIGN
10,K1
ASSIGN
9,FN3
ASSIGN
21,V71
ASSIGN
22,V72
ASSIGN
Ic
E
P2
K5
(NEXP)
TEST
9,K1
ASSIGN
TRANSFER
(INITF)
,
Ic
22,V2
ASSIGN
(INITF)
P9
G
K1
TEST
29,V71
ASSIGN
30,V72
ASSIGN
31,V2
ASSIGN
P9
G
K2
TEST
37,V71
ASSIGN
38,V72
ASSIGN
39,V2
ASSIGN
INITF
LINK
P2
FIFO

Section II


GEN
,,
,7,50,,F
SAVEVALUE
2+,K1
1,X2
ASSIGN
NOT3
2,V10
ASSIGN
3,V11
ASSIGN
4,V12
ASSIGN
IIh
IIh
5,V13
ASSIGN
MSAVEVALUE
INFO,P1,
K2,KO
MSAVEVALUE
INFO,P1
K3,KO
UNH1
(TEST1)
CH*2
G
KO
TEST
IIa
IIb
SAVEVALUE
5,P2
UNH
X5
1
UNLINK
ENTER
IId
IIi
IIc
IIi
SAVEVALUE
6,P1
BUFFER
8,X7
ASSIGN
(UNH2)
P2
E
K5
TEST
LINK
7
FIFO
UNH2
TRANSFER
,
(REQN)
IIIa

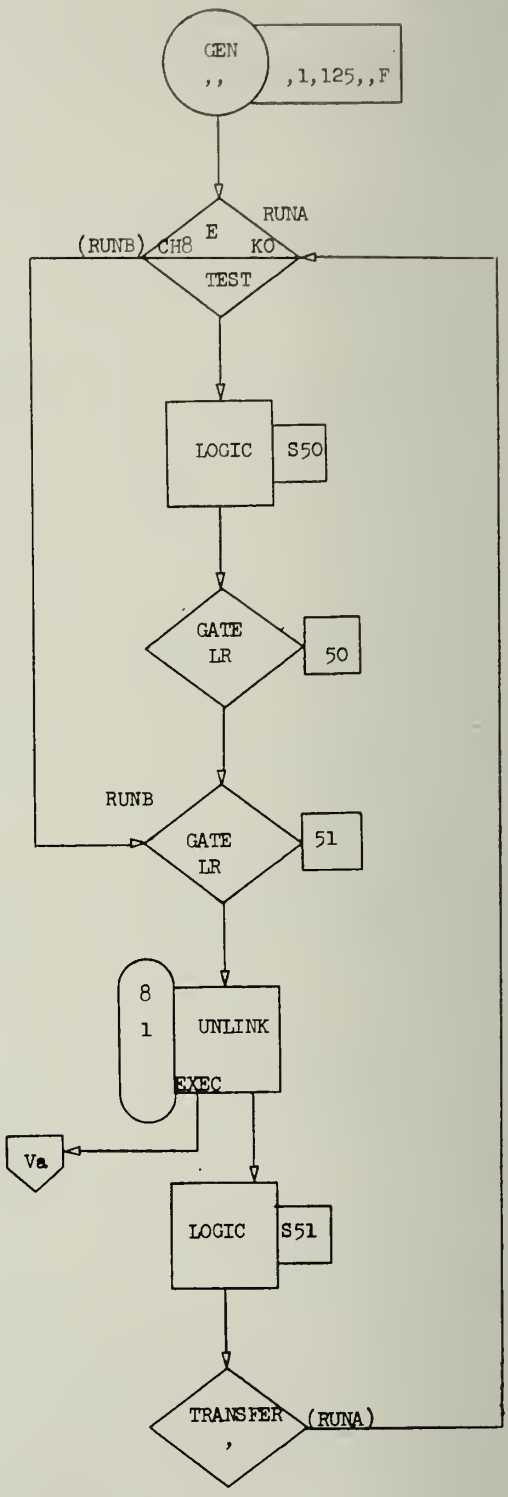

IIc
ENTER
MARK
13
48,K1
ASSIGN
8,X6
ASSIGN
MSAVEVALUE
INFO,P8
K1,P1
MSAVEVALUE
INFO,P8
K3,K1
MSAVEVALUE
INFO,P8
K6,P2
MSAVEVALUE
INFO,P8
K9,P12
SAVEVALUE
7,P1
IIf
IIf
ENT1
12,V41
ASSIGN
3,V*12
ASSIGN
MSAVEVALUE
INFO,P8
K7,P*12
4,K0
ASSIGN
12,V42
ASSIGN
5,P*12
ASSIGN
MSAVEVALUE
INFO,P8
K8,P*12
6,K0
ASSIGN
MSAVEVALUE
INFO,P8
K2,P10
IIg
IIr
E
(ENT2) P2
K5
TEST
MARK
MARK
TRANSFER
,
(ACT)
IIIb
ENT2
12,V43
ASSIGN
MSAVEVALUE
INFO,P8
K4,P*12
MARK
LINK

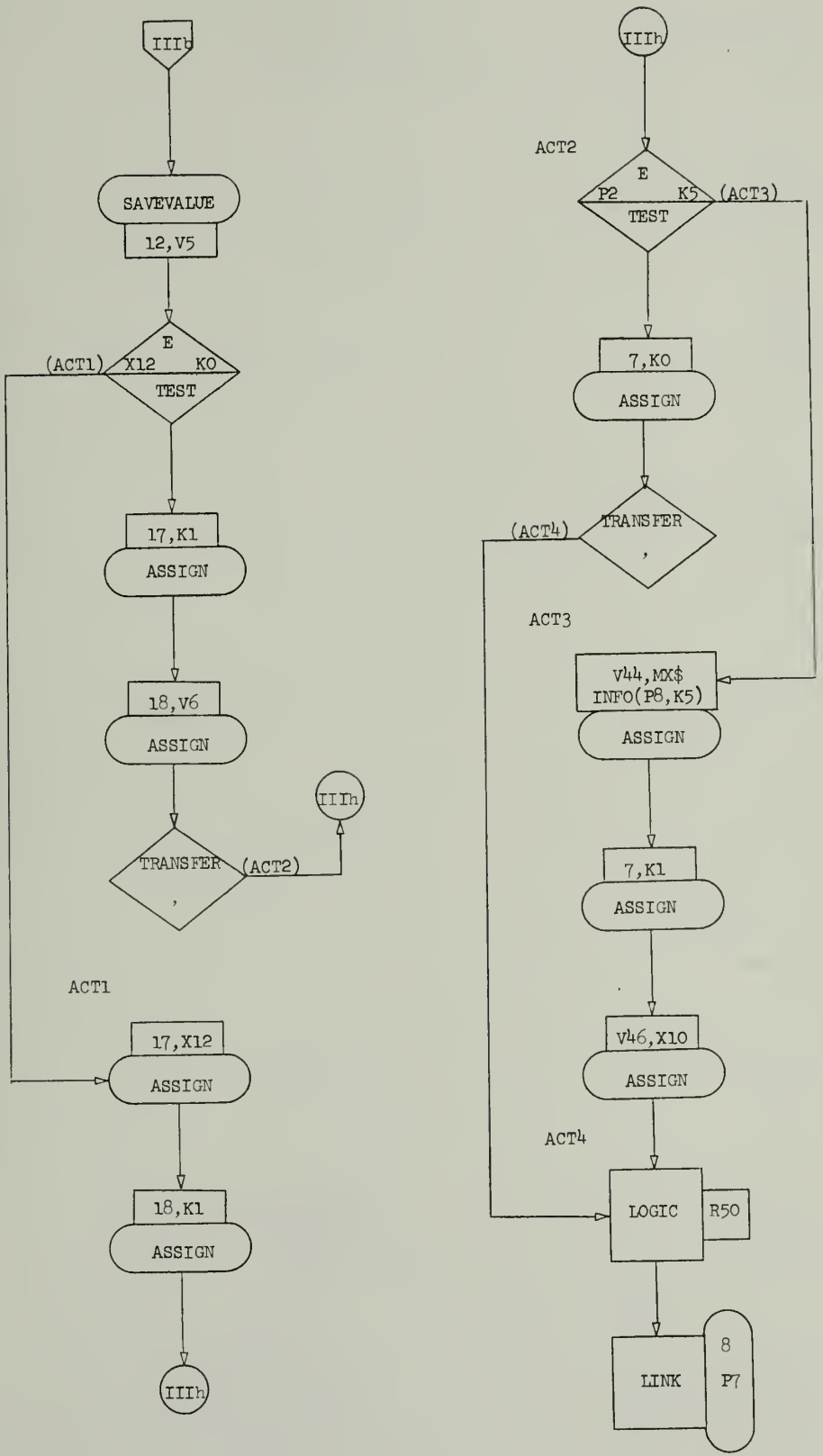

Section IIa
TEST1
P3 NE KO (BLOCK)
TEST
IIb
IIe
CH*3 G KO (TEST2)
TEST
SAVEVALUE
5,P3
TRANSFER (UNH)
,
IId
TEST2
P4 NE KO (BLOCK)
TEST
IIe
CH*4 G KO (TEST3)
TEST
SAVEVALUE
5,P4
TRANSFER (UNH)
,
IId
TEST3
P5 NE KO (BLOCK)
TEST
IIe
(BLOCK) CH*5 G KO
TEST
IIe
SAVEVALUE
5,P5
TRANSFER (UNH)
,
IId
IIe
BLOCK
LINK
6
FIFO
IIIa
SAVEVALUE
8,K1
IIIf
MX$ E REQS
CORE(1,X8) KO
TEST
IIIc
11,MX$INFO
(P1,K4)
ASSIGN
SAVEVALUE
9,X8
IIIh
Section III
IIIh
E INNER
MX$
CORE(1,X8) KO
TEST
IIIc
SAVEVALUE
8+,K1
LE
X8 X1
TEST
11
LOOP
TRANSFER (REQF)
,
MSAVEVALUE
INFO,P1
K5,X9
IIId
IIIc
REQI
SAVEVALUE
8+,K1
LE
(REQW) X8 X1
TEST
TRANSFER (REQS)
,
IIIf
REQW
LINK
9
FIFO

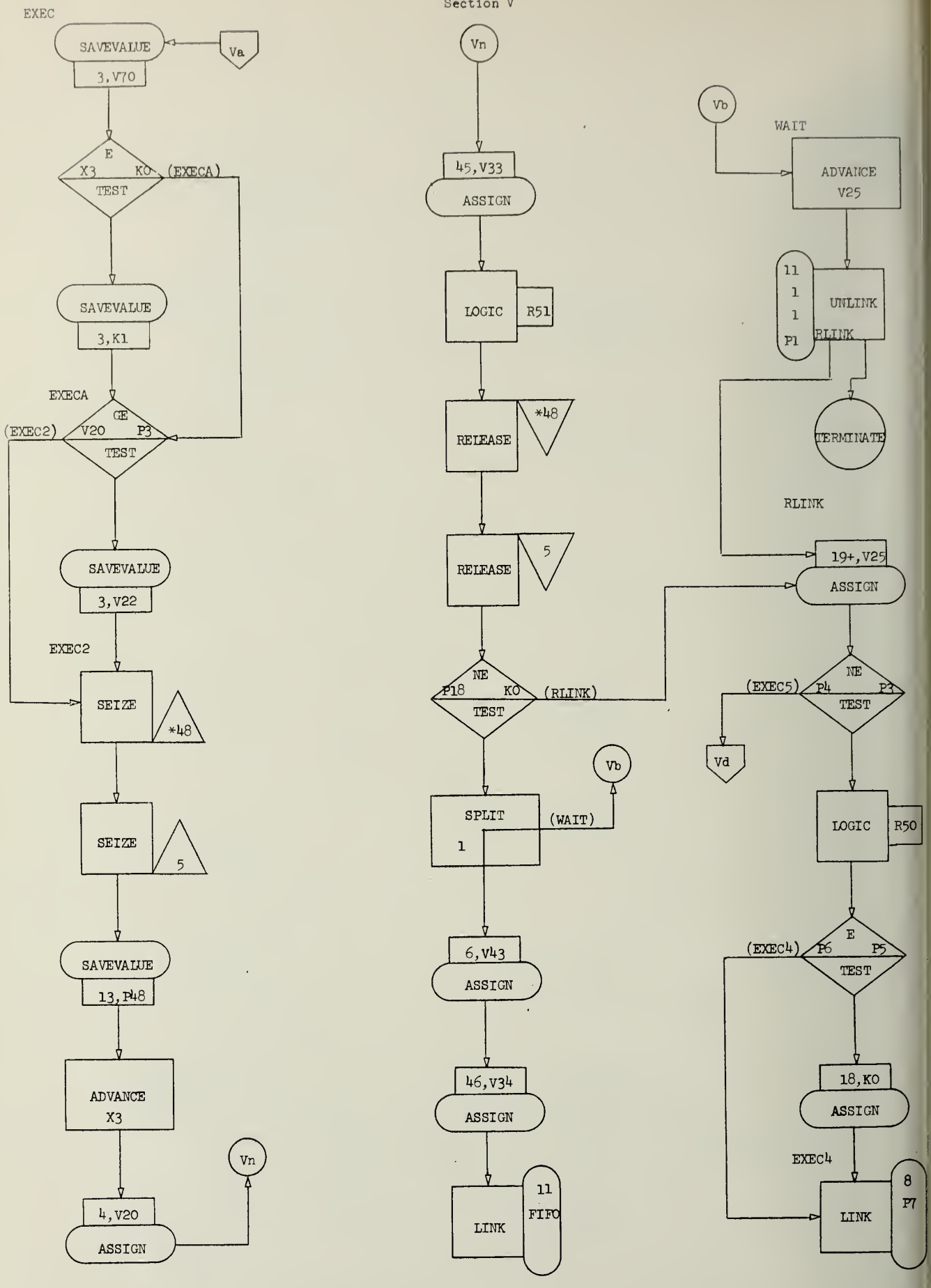


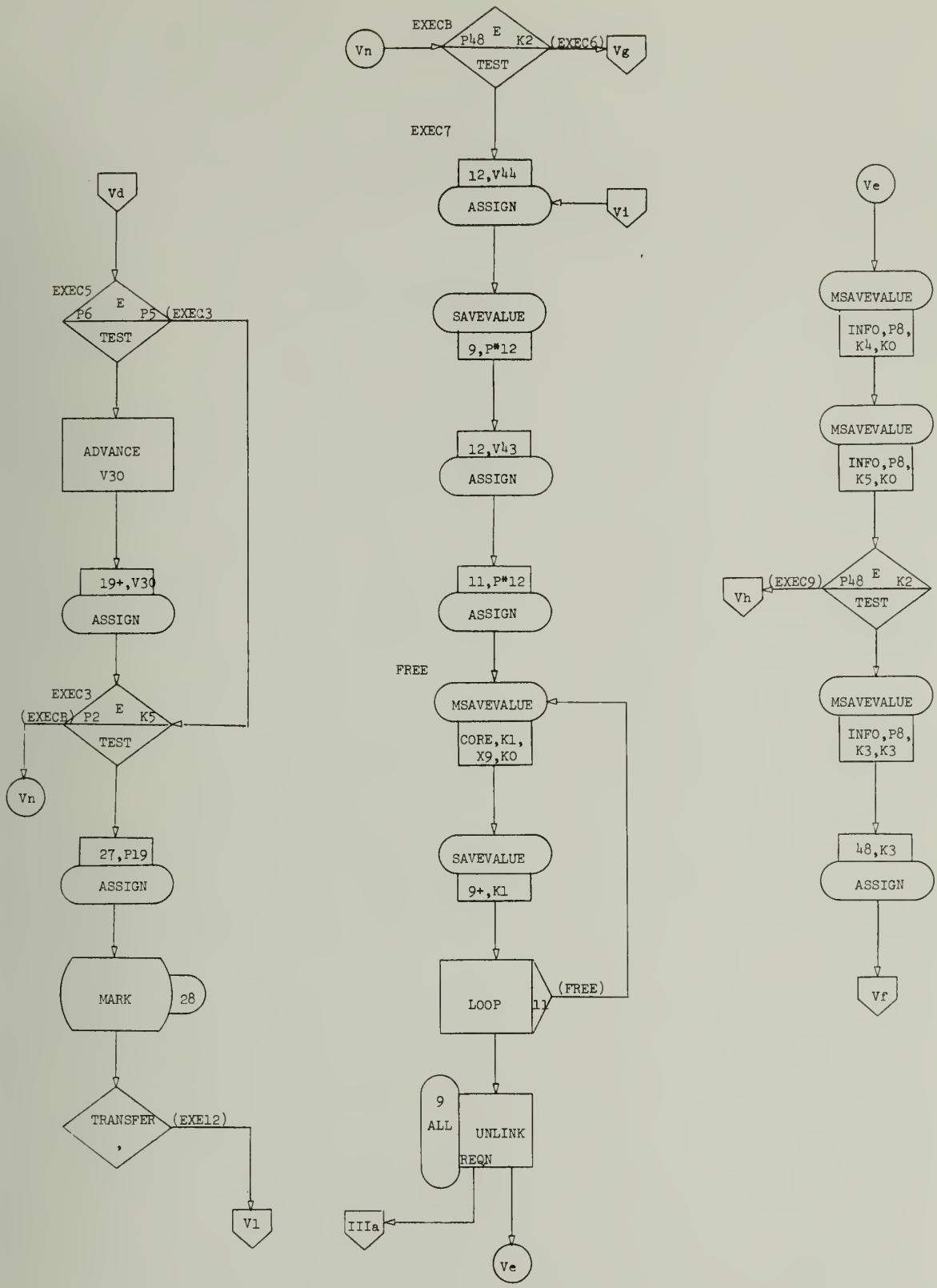

Section III (Continued)

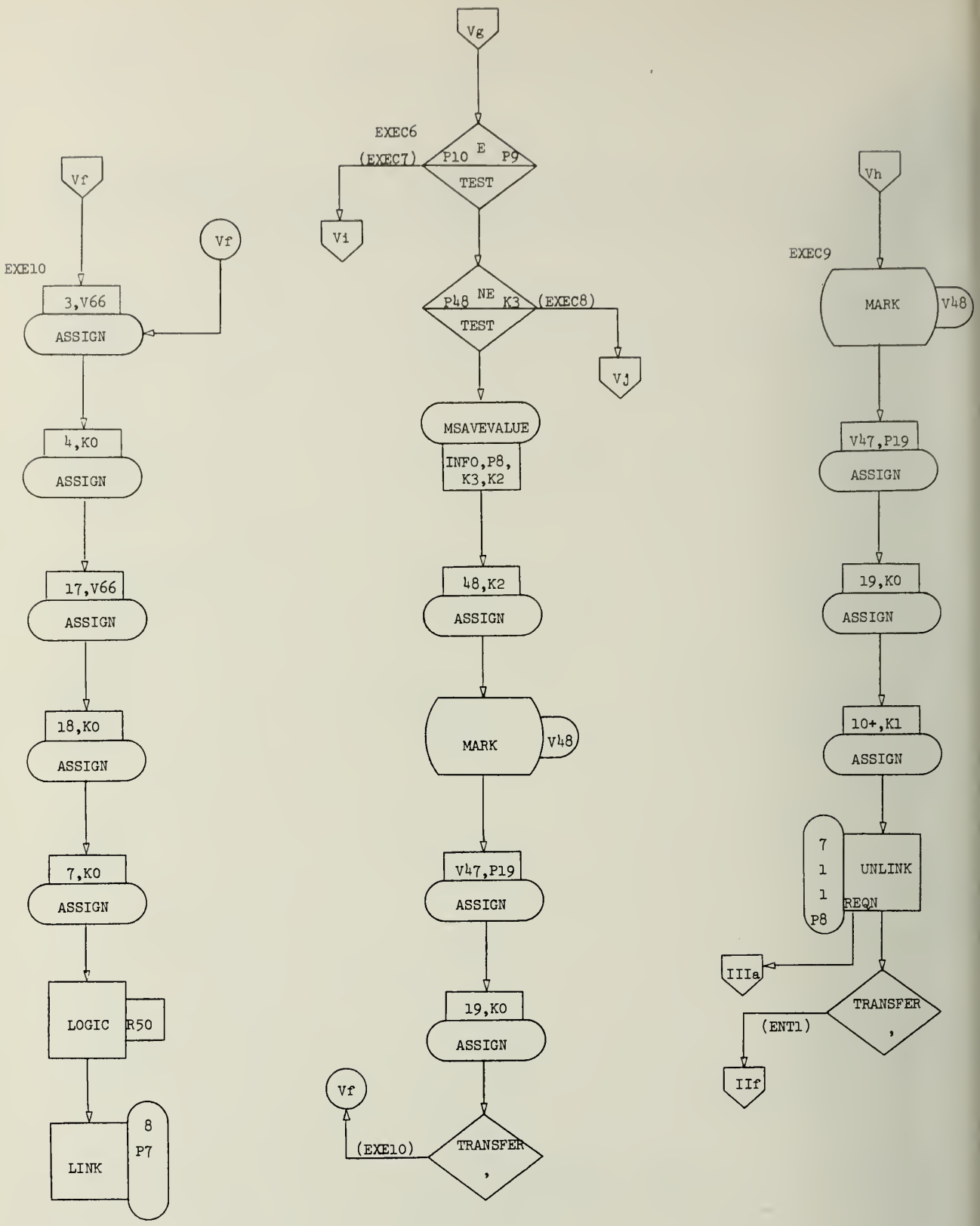
Section IV

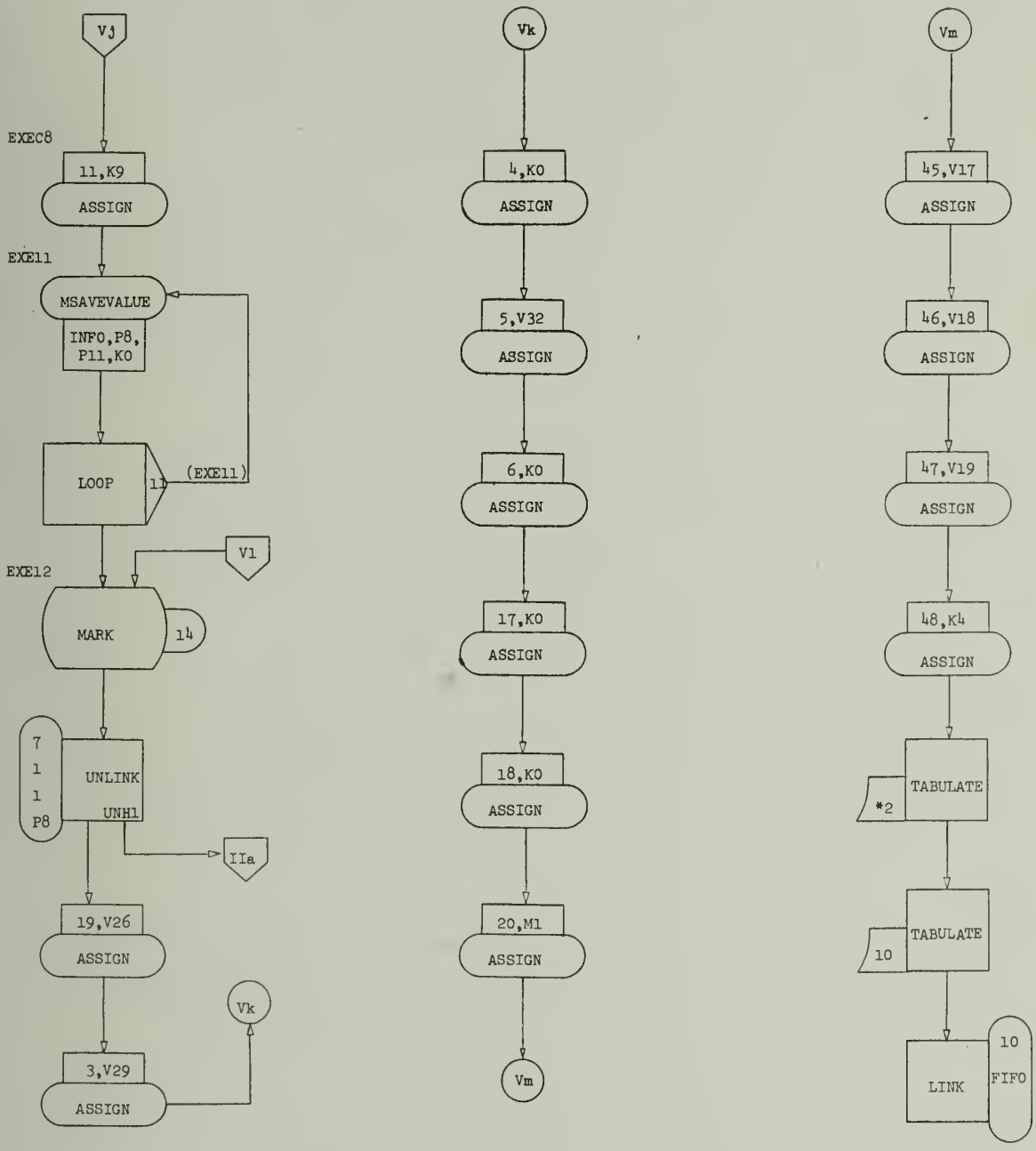



IIIb
SAVEVALUE
12,V5
E
X12 K0
TEST
(ACT1)
17,K1
ASSIGN
18,V6
ASSIGN
IIIh
TRANSFER
,
(ACT2)
ACT1
17,X12
ASSIGN
18,K1
ASSIGN
IIIh
IIIh
ACT2
E
P2 K5
TEST
(ACT3)
7,K0
ASSIGN
TRANSFER
,
(ACT4)
ACT3
V44,MX$
INFO(P8,K5)
ASSIGN
7,K1
ASSIGN
V46,X10
ASSIGN
ACT4
LOGIC
R50
LINK
8
P7

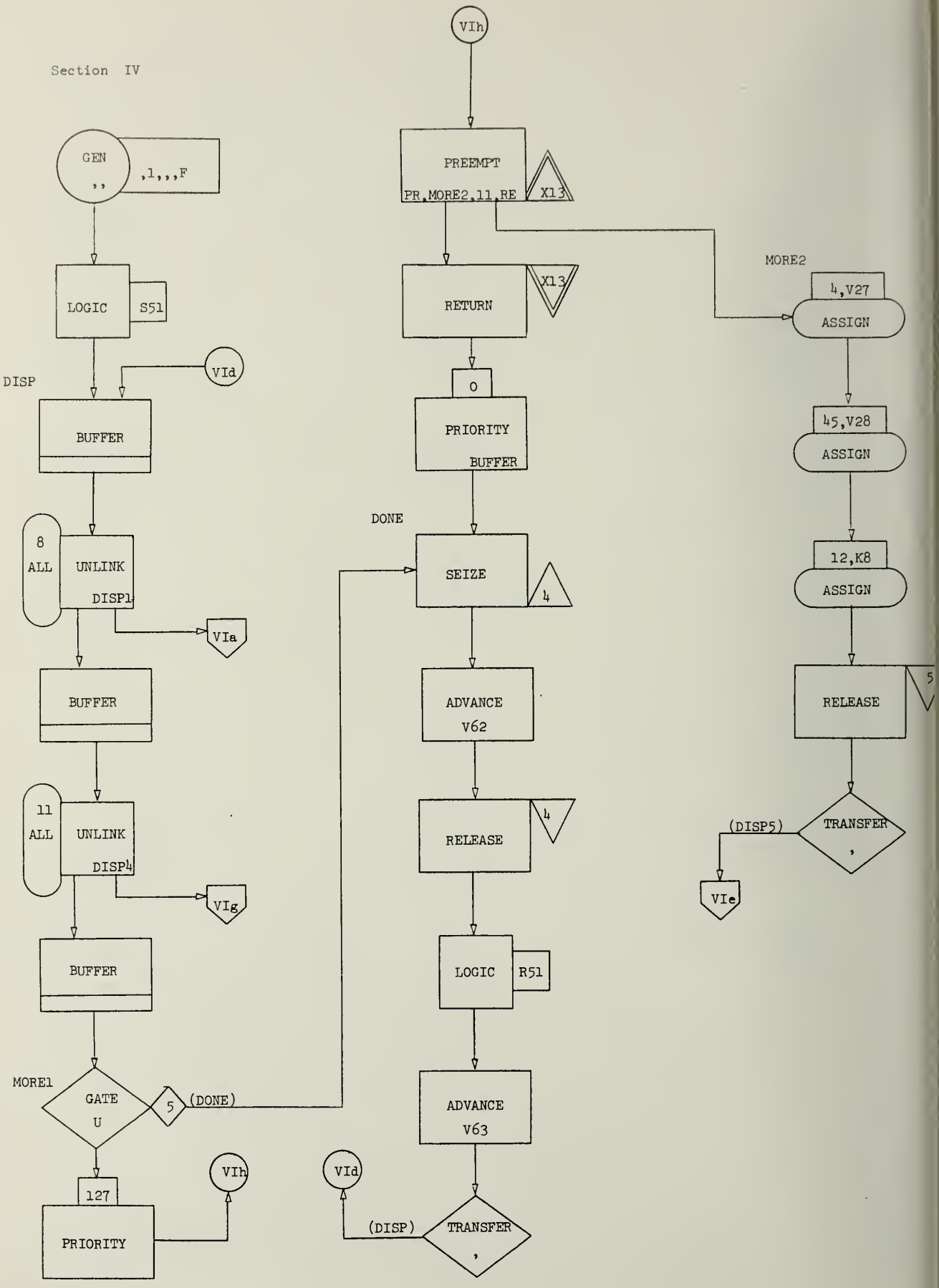

EXEC
SAVEVALUE
3,V70
Va
E
X3 KO (EXECA)
TEST
SAVEVALUE
3,K1
EXECA
GE
(EXEC2) V20 P3
TEST
SAVEVALUE
3,V22
EXEC2
SEIZE
*48
SEIZE
5
SAVEVALUE
13,P48
ADVANCE
X3
Vn
4,V20
ASSIGN
Vn
45,V33
ASSIGN
LOGIC
R51
RELEASE
*48
RELEASE
5
NE
P18 KO (RLINK)
TEST
Vb
SPLIT
1
(WAIT)
6,V43
ASSIGN
46,V34
ASSIGN
LINK
11
FIFO
Vb
WAIT
ADVANCE
V25
11
1
1
P1
UNLINK
RLINK
TERMINATE
RLINK
19+,V25
ASSIGN
NE
(EXEC5) P4 P3
TEST
Vd
LOGIC
R50
E
(EXEC4) P6 P5
TEST
18,KO
ASSIGN
EXEC4
LINK
8
P7


Vd
EXEC5
P6 E P5 (EXEC3
TEST
ADVANCE
V30
19+,V30
ASSIGN
EXEC3
(EXECB) P2 E K5
TEST
Vn
27,P19
ASSIGN
MARK
28
TRANSFER
,
(EXE12)
V1
EXECB
Vn
P48 E K2 (EXEC6)
TEST
Vg
EXEC7
12,V44
ASSIGN
Vi
SAVEVALUE
9,P*12
12,V43
ASSIGN
11,P*12
ASSIGN
FREE
MSAVEVALUE
CORE,K1,
X9,KO
SAVEVALUE
9+,K1
LOOP
11
(FREE)
9
ALL
UNLINK
REQN
IIIa
Ve
Ve
MSAVEVALUE
INFO,P8,
K4,KO
MSAVEVALUE
INFO,P8,
K5,KO
Vh
(EXEC9)
P48 E K2
TEST
MSAVEVALUE
INFO,P8,
K3,K3
48,K3
ASSIGN
Vf

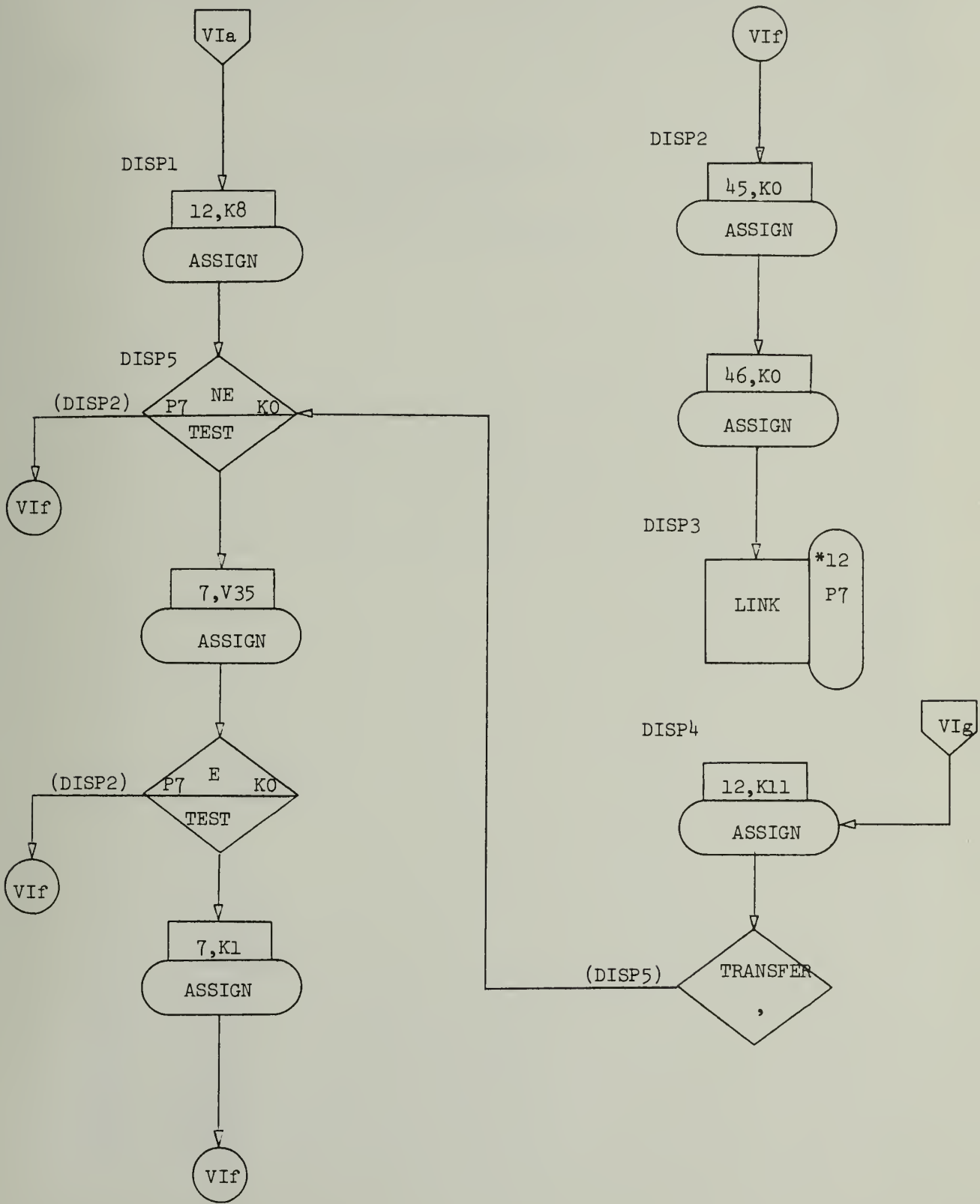

Vf
EXE10
3,V66
ASSIGN
Vf
4,K0
ASSIGN
17,V66
ASSIGN
18,K0
ASSIGN
7,K0
ASSIGN
LOGIC
R50
LINK
8
P7
Vg
EXEC6
(EXEC7)
P10 E P9
TEST
Vi
P48 NE K3
TEST
(EXEC8)
Vj
MSAVEVALUE
INFO,P8,
K3,K2
48,K2
ASSIGN
MARK
V48
V47,P19
ASSIGN
19,K0
ASSIGN
Vf
(EXE10)
TRANSFER
,
Vh
EXEC9
MARK
V48
V47,P19
ASSIGN
19,K0
ASSIGN
10+,K1
ASSIGN
7
1
1
P8
UNLINK
REQN
IIIa
TRANSFER
,
(ENT1)
IIf


Vj
EXEC8
11,K9
ASSIGN
EXE11
MSAVEVALUE
INFO,P8,
P11,K0
LOOP
11
(EXE11)
V1
EXE12
MARK
14
7
1
1
P8
UNLINK
UNH1
IIa
19,V26
ASSIGN
Vk
3,V29
ASSIGN
Vk
4,K0
ASSIGN
5,V32
ASSIGN
6,K0
ASSIGN
17,K0
ASSIGN
18,K0
ASSIGN
20,M1
ASSIGN
Vm
Vm
45,V17
ASSIGN
46,V18
ASSIGN
47,V19
ASSIGN
48,K4
ASSIGN
*2
TABULATE
10
TABULATE
LINK
10
FIFO

Section IV


GEN
,,
,1,,F
LOGIC
S51
DISP
VId
BUFFER
8
ALL
UNLINK
DISP1
VIa
BUFFER
11
ALL
UNLINK
DISP4
VIg
BUFFER
MORE1
GATE
U
5
(DONE)
127
PRIORITY
VIh
VIh
PREEMPT
PR,MORE2,11,RE
X13
RETURN
X13
0
PRIORITY
BUFFER
DONE
SEIZE
4
ADVANCE
V62
RELEASE
4
LOGIC
R51
ADVANCE
V63
VId
(DISP)
TRANSFER
,
MORE2
4,V27
ASSIGN
45,V28
ASSIGN
12,K8
ASSIGN
RELEASE
5
(DISP5)
TRANSFER
,
VIe

Section VIa


VIa
DISP1
12,K8
ASSIGN
DISP5
(DISP2)
P7
NE
K0
TEST
VIf
7,V35
ASSIGN
(DISP2)
P7
E
K0
TEST
VIf
7,K1
ASSIGN
VIf
VIf
DISP2
45,K0
ASSIGN
46,K0
ASSIGN
DISP3
LINK
*12
P7
VIg
DISP4
12,K11
ASSIGN
(DISP5)
TRANSFER
,

# APPENDIX B

```
      REALLOCATE FAC,50,STO,10,QUE,10,LOG,55,FSV,50,COM,80000,VAR,75
      REALLOCATE TA8,10,8LG,500,FUN,20

8LOCK                                                                                                  CARD
NUM8ER   *LOC    OPERATION  A,8,C,D,E,F,G              COMMENTS                                        NUM8ER
         *                                                                                                1
                 SIMULATE                                                                                 2
         *                                                                                                3
          1      TA8LE      P20,0,25000,15                                                                4
          2      TA8LE      P20,0,25000,15                                                                5
          3      TA8LE      P20,0,40000,15                                                                6
          4      TA8LE      P20,0,20000,15                                                                7
          5      TA8LE      P20,0,2500,15                                                                 8
          10     TA8LE      P20,0,25000,15                                                                9
         *                                                                                               10
          CORE   MATRIX     X,1,201                                                                      11
          INFO   MATRIX     X,7,9                                                                        12
          IOREQ  MATRIX     X,1,5                                                                        13
          TIME   MATRIX     X,1,5                                                                        14
         *                                                                                               15
                 INITIAL    MX$CORE(1,201),-1                                                            16
                 INITIAL    MX$TIME(1,1),1606/MX$TIME(1,2),4935                                          17
                 INITIAL    MX$TIME(1,3),7096/MX$TIME(1,4),8420                                          18
                 INITIAL    MX$IOREQ(1,1),77/MX$IOREQ(1,2),368                                           19
                 INITIAL    MX$IOREQ(1,3),965/MX$IOREQ(1,4),1224                                         20
                 INITIAL    MX$TIME(1,5),1000/MX$IOREQ(1,5),50                                           21
                 INITIAL    MX$INFO(2-4,2),1/MX$INFO(5-6,2),2/MX$INFO(7,2),3                             22
                 INITIAL    MX$INFO(2-4,3),0/MX$INFO(5-6,3),0/MX$INFO(7,3),2                             23
                 INITIAL    MX$INFO(1,2),5                                                               24
                 INITIAL    X1,201                                                                       25
         *                                                                                               26
          1      FUNCTION   RN2,D4                                                                       27
         .6,1/.9,2/.93,3/1,4                                                                             28
          2      FUNCTION   RN2,D5                                                                       29
         .21,1/.49,2/.53,3/.61,4/1.0,5                                                                   30
          3      FUNCTION   RN2,D3                                                                       31
         .05,2/.20,3/1,1                                                                                 32
          EXP    FUNCTION   RN4,C24                                                                      33
         0,.100/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38                           34
         .8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2                              35
         .97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8                                        36
          EXP1   FUNCTION   RN5,C24                                                                      37
         0,.100/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38                           38
         .8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2                              39
         .97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8                                        40
          EXP2   FUNCTION   RN3,C24                                                                      41
         0,.100/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38                           42
         .8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2                              43
         .97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8                                        44
         *                                                                                               45
          1      VARIA8LE   500000                                                                       46
          2      VARIA8LE   RN6/10                                                                       47
          5      VARIA8LE   P3/P5                                                                        48
          6      VARIA8LE   P5/P3                                                                        49
          7      VARIA8LE   P1                                                                           50
          3      VARIA8LE   P8                                                                           51
          4      VARIA8LE   10+P8                                                                        52
          8      VARIA8LE   K50                                                                          53
          9      VARIA8LE   CH8+CH11                                                                     54
          10     VARIA8LE   MX$INFO(X2,K2)                                                               55
          11     VARIA8LE   MX$INFO(X2,K3)                                                               56
          12     VARIA8LE   MX$INFO(X2,K4)                                                               57
          13     VARIA8LE   MX$INFO(X2,K5)                                                               58
          17     VARIA8LE   P28-P27-P26-P21                                                              59
          18     VARIA8LE   P36-P35-P34-P29                                                              60
          19     VARIA8LE   P44-P43-P42-P37                                                              61
          20     VARIA8LE   P4+X3                                                                        62
          22     VARIA8LE   P3-P4                                                                        63
          23     VARIA8LE   P6+P18                                                                       64
          24     VARIA8LE   P4/P6                                                                        65
          25     VARIA8LE   P18*V61                                                                      66
          26     VARIA8LE   P27+P35+P43                                                                  67
          27     VARIA8LE   P4+(X3-P11)                                                                  68
          28     VARIA8LE   P45+(X3-P11)                                                                 69
          29     VARIA8LE   P21+P29+P37                                                                  70
          30     VARIA8LE   (P5-P6)*V61                                                                  71
          31     VARIA8LE   P15-P14-P4-P19                                                               72
          32     VARIA8LE   P22+P30+P38                                                                  73
          33     VARIA8LE   P45+X3                                                                       74
          34     VARIA8LE   P46+P18                                                                      75
          35     VARIA8LE   P45/P46                                                                      76
          41     VARIA8LE   13+P10*8                                                                     77
          42     VARIA8LE   14+P10*8                                                                     78
          43     VARIA8LE   15+P10*8                                                                     79
          44     VARIA8LE   16+P10*8                                                                     80
          45     VARIA8LE   17+P10*8                                                                     81
          46     VARIA8LE   18+P10*8                                                                     82
          47     VARIA8LE   19+P10*8                                                                     83
          48     VARIA8LE   20+P10*8                                                                     84
```

```
 60     VARIABLE   K15000                                                  85
 61     VARIABLE   K100                                                   86
 62     VARIABLE   K50                                                    87
 63     VARIABLE   K2000                                                  88
 64     VARIABLE   K15                                                    89
 65     VARIABLE   K1800000                                               90
 66     VARIABLE   K2500                                                  91
 70     FVARIABLE  P17*FN$EXP2                                            92
 71     FVARIABLE  (MX$TIME(1,P2))*FN$EXP                                 93
 72     FVARIABLE  (MX$IOREQ(1,P2))*FN$EXP                                94
*                                                                        95
*                    **   SECTION I   **                                96
*                                                                        97
*    CREATE JOB STREAM              (JOBS)                               98
*                                                                        99
*    INITIALIZE STARTING JOBS ON THE QUEUE                              100
1    NOT1  GENERATE   ,,,V64,100,48,F                                   101
2          ASSIGN     2,FN1            ASSIGN CLASS FOR INITIAL JOBS    102
3          TRANSFER   ,BEGIN                                            103
*                                                                       104
4    NOT2  GENERATE   V60,FN$EXP1,,,100,48,F                            105
5          ASSIGN     2,FN2            ASSIGN JOB CLASS                 106
6    BEGIN MARK       12               MARK CREATION TIME TO PASS TO INFO107
7          UNLINK     6,UNH1,ALL       FREE WAITING INITIATORS          108
8          SAVEVALUE  4+,K1            INCREMENT JOB NUMBER COUNTER     109
9          ASSIGN     1,X4             ASSIGN JOB NUMBER                110
10         ASSIGN     10,K1            STEP COUNTER TO 1                111
11         ASSIGN     9,FN3            ASSIGN NUMBER OF STEPS           112
12         ASSIGN     21,V71           STEP 1 TIME                      113
13         ASSIGN     22,V72           STEP 1 I/O REQUESTS              114
14         TEST E     P2,K5,NEXP       IS THIS AN EXPRESS JOB?          115
15         ASSIGN     9,K1             EXPRESS JOBS CAN HAVE ONLY ONE STEP116
16         TRANSFER   ,INITF                                            117
17   NEXP  ASSIGN     23,V2            STEP 1 CORE REQUEST              118
18         TEST G     P9,K1,INITF      ANY MORE STEPS?                  119
19         ASSIGN     29,V71           STEP 2 TIME                      120
20         ASSIGN     30,V72           STEP 2 I/O REQUESTS              121
21         ASSIGN     31,V2            STEP 2 CORE REQUEST              122
22         TEST G     P9,K2,INITF      ANY MORE STEPS?                  123
23         ASSIGN     37,V71           STEP 3 TIME                      124
24         ASSIGN     38,V72           STEP 3 I/O REQUESTS              125
25         ASSIGN     39,V2            STEP 3 CORE REQUEST              126
26   INITF LINK       P2,FIFO          PUT JOB ON "HASP SPOOL"          127
*                                                                       128
*                  **   SECTION II   **                                129
*                                                                       130
*    CREATE SEVEN INITIATOR/TERMINATORS     (INITIATORS)               131
*                                                                       132
27         GENERATE   ,,,7,50,,F                                        133
28         SAVEVALUE  2+,K1            COUNTER                          134
29         ASSIGN     1,X2             ASSIGN INITIATOR NUMBER          135
30   NOT3  ASSIGN     2,V10            ASSIGN INITIATOR PRIMARY CLASS   136
31         ASSIGN     3,V11            ASSIGN INITIATOR SECONDARY CLASS 137
32         ASSIGN     4,V12            ASSIGN INITIATOR THIRD CLASS     138
33         ASSIGN     5,V13            ASSIGN INITIATOR FOURTH CLASS    139
34         MSAVEVALUE INFO,P1,K2,K0    ZERO OUT TEMP INFO IN "INFO"     140
35         MSAVEVALUE INFO,P1,K3,K0    ZERO OUT TEMP INFO IN "INFO"     141
36   UNH1  TEST G     CH*2,K0,TEST1    ANYTHING ON THE SPOOL?           142
37         SAVEVALUE  5,P2             INDICATE WHICH CLASS SELECTED    143
38   UNH   UNLINK     X5,ENTER,1       UNLINK JOB THAT JUST GOT INITIATOR144
39         SAVEVALUE  6,P1             PASS INITIATOR NUMBER TO JOB     145
40         BUFFER     MAKE APPRCPRIATE JOB ASSIGNMENTS                 146
41         ASSIGN     8,X7             PUT CORRESPONDING JOB NO. IN INIT147
42         TEST E     P2,K5,UNH2       IS THIS AN EXPRESS JOB?          148
43         LINK       7,FIFO           SKIP CORE ALLOCATION ROUTINE     149
44   UNH2  TRANSFER   ,REQN            GET CORE ALLOCATION              150
*                                                                       151
*                  **   SECTION II-A   **                              152
*                                                                       153
*   CHECK ALTERNATE CLASS DEFINITIONS       (INITIATORS)               154
*                                                                       155
45   TEST1 TEST NE    P3,K0,BLOCK      CHECK FOR SECONDARY CLASS DEFINITION156
46         TEST G     CH*3,K0,TEST2    ANY JOBS ON THIS CHAIN?          157
47         SAVEVALUE  5,P3             INDICATE WHICH CLASS SELECTED    158
48         TRANSFER   ,UNH                                              159
49   TEST2 TEST NE    P4,K0,BLOCK      CHECK FOR THIRD CLASS DEFINITION 160
50         TEST G     CH*4,K0,TEST3    ANY JOBS ON THIS CHAIN?          161
51         SAVEVALUE  5,P4             INDICATE WHICH CLASS SELECTED    162
52         TRANSFER   ,UNH                                              163
53   TEST3 TEST NE    P5,K0,BLOCK      CHECK LAST CLASS DEFINITION      164
54         TEST G     CH*5,K0,BLOCK    ANY JOBS ON THIS CHAIN?          165
55         SAVEVALUE  5,P5             INDICATE WHICH CLASS SELECTED    166
56         TRANSFER   ,UNH                                              167
57   BLOCK LINK       6,FIFO           PUT BLOCKED INITIATOR IN WAIT STATE168
*                                                                       169
```

```
*                **   SECTION II-8   **                                    170
*                                                                          171
*    ENTER JOB INFO INTO "INFO"              (JOBS)                        172
*                                                                          173
58   ENTER MARK       13              ENTER TIME REMOVED FROM SPOOL        174
59         ASSIGN     48,K1           INITIALIZE STATUS COUNTER            175
60         ASSIGN     8,X6            ASSIGN INITIATOR NUMBER TO JOB       176
61         MSAVEVALUE INFO,P8,K1,P1   ENTER JOB NUMBER                     177
62         MSAVEVALUE INFO,P8,K3,K1   SEND STATUS TO INFO                  178
63         MSAVEVALUE INFO,P8,K6,P2   ENTER CLASS OF THIS JOB              179
64         MSAVEVALUE INFO,P8,K9,P12  PUT JOB CREATION TIME IN INFO        180
65         SAVEVALUE  7,P1            SEND JOB NO. TO INITIATOR            181
66   ENT1  ASSIGN     12,V41                                               182
67         ASSIGN     3,P*12          ASSIGN STEP EXEC TIME                183
68         MSAVEVALUE INFO,P8,K7,P*12 ENTER STEP TIME                      184
69         ASSIGN     4,K0            ASSIGN ELAPSED STEP TIME             185
70         ASSIGN     12,V42                                               186
71         ASSIGN     5,P*12          ASSIGN STEP I/O REQUESTS             187
72         MSAVEVALUE INFO,P8,K8,P*12 ENTER STEP I/O REQUESTS              188
73         ASSIGN     6,K0            ASSIGN ELAPSED STEP I/O REQUESTS     189
74         MSAVEVALUE INFO,P8,K2,P10  SEND STEP NUMBER TO INFO             190
75         TEST E     P2,K5,ENT2      IS THIS AN EXPRESS JOB?              191
76         MARK       25              MARK STEP START                      192
77         MARK       26              MARK TIME  (EXPRESS DOESN'T WAIT)    193
78         TRANSFER   ,ACT            GO DIRECTLY TO READY QUEUE           194
79   ENT2  ASSIGN     12,V43                                               195
80         MSAVEVALUE INFO,P8,K4,P*12 ENTER STEP CORE REQUEST              196
81         MARK       V45             MARK STEP START                      197
82         LINK       13,FIFO                                              198
*                                                                          199
*                    **   SECTION III   **                                200
*                                                                          201
*   CORE ALLOCATION ROUTINE                  (INITIATORS)                  202
*                                                                          203
83   REQN  SAVEVALUE  8,K1            INITIALIZE BASE COUNTER              204
84   REQS  TEST E     MX$CORE(1,X8),K0,REQI     IS THIS A FREE BLOCK?      205
85         ASSIGN     11,MX$INFO(P1,K4)    INITIALIZE CORE REQUEST (SIZE)  206
86         SAVEVALUE  9,X8             STORE TEMPORARY BASE                 207
87   INNER TEST E     MX$CORE(1,X8),K0,REQI     IS THIS A FREE BLOCK?      208
88         SAVEVALUE  8+,K1            INCREMENT CORE LOCATION              209
89         TEST LE    X8,X1,REQW       IS MORE CORE AVAILABLE               210
90         LOOP       11,INNER                                             211
91         TRANSFER   ,REQF            REQUEST FULFILLED                    212
92   REQI  SAVEVALUE  8+,K1            INCREMENT CORE LOCATION              213
93         TEST LE    X8,X1,REQW       IS MORE CORE AVAILABLE               214
94         TRANSFER   ,REQS                                                215
95   REQW  LINK       9,FIFO           WAIT FOR CORE TO BECOME AVAILABLE    216
96   REQF  MSAVEVALUE INFO,P1,K5,X9     ENTER BASE ADDRESS OF ALLOCATION    217
97         ASSIGN     11,MX$INFO(P1,K4)    INITIALIZE CORE REQUEST (SIZE)  218
98   REQA  MSAVEVALUE CORE,K1,X9,MX$INFO(P1,K1) PUT JOB NO. IN CORE MAP     219
99         SAVEVALUE  9+,K1            INCREMENT CORE LOCATION              220
100        LOOP       11,REQA                                              221
101        MARK       9                TIME CORE REQUEST FILLED            222
102        SAVEVALUE  10,P9            PASS THIS TIME TO JOB                223
103  REQT  UNLINK     13,ACT,1,8,P1    PREPARE JOB TO RUN                   224
104        LINK       7,FIFO           HOLD COMPLETED REQUESTS TEMPORARILY  225
*                                                                          226
*                    **   SECTION III-A   **                              227
*                                                                          228
*    PUT JOB ON RUN-STATUS BY DISPATCHING PRIORITY    (JOBS)               229
*                                                                          230
105  ACT   SAVEVALUE  12,V5            DEFINE TIME BETWEEN INTERRUPT FOR I/O 231
106        TEST E     X12,K0,ACT1      IS THE RATIO ZERO?                   232
107        ASSIGN     17,K1            1 IS THE MINIMUM RATIO               233
108        ASSIGN     18,V6            NUMBER OF I/O REQ. IN THIS MINIMUM   234
109        TRANSFER   ,ACT2                                                 235
110  ACT1  ASSIGN     17,X12           ASSIGN THIS RATIO                    236
111        ASSIGN     18,K1            NUMBER OF I/O IN THE SLICE EQUAL 1   237
112  ACT2  TEST E     P2,K5,ACT3       IS THIS AN EXPRESS JOB?              238
113        ASSIGN     7,K0             EXPRESS JOB GETS TOP PRIORITY        239
114        TRANSFER   ,ACT4                                                 240
115  ACT3  ASSIGN     V44,MX$INFO(P8,K5)   ASSIGN STEP ITS CORE ALLOCATION 241
116        ASSIGN     7,K1             ASSIGN INITIAL DISPATCHING PRIORITY  242
117        ASSIGN     V46,X10          ASSIGN TIME CORE REQUEST FILLED      243
118  ACT4  LOGIC R    50               SIGNAL SOMETHING ON READY QUEUE      244
119        LINK       8,P7             PUT JOB ON "READY QUEUE"             245
*                                                                          246
*                   **   SECTION IV   **                                  247
*                                                                          248
*     CONTROL SECTION         (ONE CONTROL TRANSACTION)                    249
*                                                                          250
120        GENERATE   ,,,1,125,,F      GENERATE ONE CONTROL TRANSACTION     251
121  RUNA  TEST E     CH8,K0,RUNB      ANYTHING READY TO RUN?               252
122        LOGIC S    50               WAIT FOR A READY JOB                 253
123        GATE LR    50                                                    254
124  RUN8  GATE LR    51               LIMIT CPU TO ONE TASK                255
125        UNLINK     8,EXEC,1         SEND FIRST JOB ON READY QUEUE TO EXEC. 256
126        LOGIC S    51               WAIT FOR JOB TO RUN                  257
127        TRANSFER   ,RUNA                                                 258
```

```
*
*                    **   SECTION V   **
*
*     EXECUTE JOB                              (JOBS)
*
EXEC   SAVEVALUE  3,V70             ASSIGN SLICE
       TEST E     X3,K0,EXECA       IS THE NEW SLICE ZERO?
       SAVEVLAUE  3,K1              ASSIGN MINIMUM SLICE
EXECA  TEST GE    V20,P3,EXEC2      IS STEP FINISHED?
       SAVEVALUE  3,V22             ASSIGN SLICE TO EXECUTE TO COMPLETE
EXEC2  SEIZE      *48               TAKE THE CPU
       SEIZE      5                 ACCOUNT FOR ANY USE OF THE CPU
       SAVEVALUE  13,P48            PASS THIS FACILITY TO DISPATCHER
       ADVANCE    X3                ADVANCE THE SLICE
       ASSIGN     4,V20             UPDATE ELAPSED TIME
       ASSIGN     45,V33            UPDATE TIME IN THE INTERVAL
       LOGIC R    51                FREE THE SCHEDULER
       RELEASE    *48               FREE THE CPU
       RELEASE    5                 INDICATE CPU FREE
       TEST NE    P18,K0,RLINK      ANY MORE I/O REQUESTS TO ISSUE?
       SPLIT      1,WAIT            CREATE DUMMY TRANSACTION
       ASSIGN     6,V23             UPDATE ELAPSED I/O REQUESTS
       ASSIGN     46,V34            UPDATE I/O IN INTERVAL
       LINK       11,FIFO           PUT JOB IN WAIT STATE
WAIT   ADVANCE    V25               ADVANCE FOR I/O REQUESTS
       UNLINK     11,RLINK,1,1,P1   RE-ACTIVATE WAITING JOB
       TERMINATE                     KILL THE DUMMY TRANSACTION
RLINK  ASSIGN     19+,V25           UPDATE I/O REQUESTS ISSUED
       TEST NE    P4,P3,EXEC5       DID STEP COMPLETE?
       LOGIC R    50                SIGNAL SOMETHING ON READY QUEUE
       TEST E     P6,P5,EXEC4       ARE ALL I/O REQUESTS FINISHED?
       ASSIGN     18,K0             NO MORE I/O REQUESTS
EXEC4  LINK       8,P7              RETURN TO READY QUEUE
EXEC5  TEST E     P6,P5,EXEC3
       ADVANCE    V30
       ASSIGN     19+,V30
EXEC3  TEST E     P2,K5,EXEC8
       ASSIGN     27,P19
       MARK       28
       TRANSFER   ,EXE12
EXEC8  TEST E     P48,K2,EXEC6      DID JOB JUST GO O'HEAD IN CORE?
EXEC7  ASSIGN     12,V44
       SAVEVALUE  9,P*12            RE-INITIALIZE STEP CORE BASE
       ASSIGN     12,V43
       ASSIGN     11,P*12           RE-INITIALIZE STEP CORE LENGTH
FREE   MSAVEVALUE CORE,K1,X9,K0     DE-ALLOCATE THIS PIECE OF CORE
       SAVEVALUE  9+,K1             INCREMENT CORE LOCATION
       LOOP       11,FREE           RESET CORE MAP
       UNLINK     9,REQN,ALL        FREE JOBS WAITING FOR CORE
       MSAVEVALUE INFO,P8,K4,K0     RESET CORE INFORMATION IN INFO
       MSAVEVALUE INFO,P8,K5,K0     RESET CORE INFORMATION IN INFO
       TEST E     P48,K2,EXEC9      DID JOB JUST GO O'HEAD IN CORE?
NOT5   MSAVEVALUE INFO,P8,K3,K3     SET OUT OF CORE OVERHEAD INDICATOR
       ASSIGN     48,K3             SET INDICATOR TO O'HEAD OUT OF CORE
EXE10  ASSIGN     3,V66             RESET EXEC TIME
       ASSIGN     4,K0              RESET ELAPSED TIME
       ASSIGN     17,V66            SET MINIMUM TIME SLICE
       ASSIGN     18,K0             I/O REQUESTS TO ISSUE
       ASSIGN     7,K0              ASSIGN TOP PRIORITY
       LOGIC R    50                SIGNAL SOMETHING ON READY QUEUE
       LINK       8,P7              PUT BACK ON READY QUEUE
EXEC6  TEST E     P10,P9,EXEC7      ARE ALL STEPS COMPLETE?
       TEST NE    P48,K3,EXEC8      DID JUST COMPLETE O'HEAD OUT OF CORE
NOT4   MSAVEVALUE INFO,P8,K3,K2     SET OVERHEAD INDICATOR
       ASSIGN     48,K2             SET INDICATOR TO O'HEAD IN CORE
       MARK       V48               MARK STEP END
       ASSIGN     V47,P19           ENTER STEP I/O REQUESTS
       ASSIGN     19,K0             RESET I/O COUNTER
       TRANSFER   ,EXE10
EXEC9  MARK       V48               MARK STEP END
       ASSIGN     V47,P19           STEP WAIT TIME
       ASSIGN     19,K0             RESET WAIT COUNTER
       ASSIGN     10+,K1            INCREMENT STEP COUNTER
       UNLINK     7,REQN,1,1,P8     AWAKE THIS INITIATOR
       TRANSFER   ,ENT1
EXEC8  ASSIGN     11,K9             NUMBER OF ELEMENTS IN ROW OF INFO
EXE11  MSAVEVALUE INFO,P8,P11,K0    CLEAR THIS ROW OF INFO
       LOOP       11,EXE11
EXE12  MARK       14                ENTER COMPLETION TIME
       UNLINK     7,UNH1,1,1,P8     FREE INTITIATOR
       ASSIGN     19,V26            ENTER TOTAL WAIT TIME
       ASSIGN     3,V29             ENTER TOTAL EXEC TIME
```

```
205             ASSIGN      4,K0            CLEAR ELAPSED TIME                          341
206             ASSIGN      5,V32           ENTER TOTAL I/O REQUESTS                    342
207             ASSIGN      6,K0            CLEAR ELAPSED I/O REQUESTS                  343
208             ASSIGN      17,K0           ZERO OUT TEMPORARY INFORMATION              344
209             ASSIGN      18,K0           ZERO OUT TEMPORARY INFORMATION              345
210             ASSIGN      20,M1           ENTER TURNAROUND TIME                       346
211             ASSIGN      45,K0           CLEAR INTERVAL RECORDER                     347
212             ASSIGN      48,K4           MARK JOB COMPLETEION                        348
213             ASSIGN      45,V17          WAIT TIME ON READY QUEUE STEP 1             349
214             ASSIGN      46,V18          WAIT TIME ON READY QUEUE STEP 2             350
215             ASSIGN      47,V19          WAIT TIME ON READY QUEUE STEP 3             351
216             TABULATE    *2              TABULATE TURNAROUND BY CLASS                352
217             TABULATE    10              TABULATE TURNAROUND FOR OVERALL SYSTEM      353
218             LINK        10,FIFO         PUT JOB ON COMPLETED CHAIN                  354
            *                                                                           355
            *                 **   SECTION VI   **                                     356
            *                                                                           357
            *     DISPATCHING SECTION       (DISPATCHER)                              358
            *                                                                           359
219             GENERATE    ,,,1,,,F        CREATE DISPATCHER                           360
220         DISP  LOGIC S     51            BLOCK THE SCHEDULER                         361
221               LOGIC R     50            INDICATE SOMETHING ON READY QUEUE           362
222               BUFFER      LET ALL TRANSACTIONS REACH THEIR GOAL                     363
223               UNLINK      8,DISP1,ALL   REORDER READY QUEUE                         364
224               BUFFER                                                              365
225               UNLINK      11,DISP4,ALL  REORDER I/O WAITING QUEUE                   366
226               BUFFER                                                              367
227         MORE1 GATE U      5,DONE        IS THERE A JOB IN THE CPU?                  368
228               PRIORITY    127           GIVE DISPATCHER HIGH PRIORITY               369
229               PREEMPT     X13,PR,MORE2,11,RE   INTERRUPT JOB IN THE CPU            370
230               RETURN      X13           RELEASE THE CPU                             371
231               PRIORITY    0,BUFFER      REASSIGN THIS JOBS PRIORITY                 372
232         DONE  SEIZE       4             TAKE THE CPU                                373
233               ADVANCE     V62           EXECUTE DISPATCHER OVERHEAD                 374
234               RELEASE     4             RELEASE THE CPU                             375
235               LOGIC R     51            FREE THE SCHEDULER                          376
236               ADVANCE     V63           DISPATCHER TO SLEEP FOR 2 SECONDS           377
237               TRANSFER    ,DISP         REPEAT RE-ORDERING PROCESS                  378
238         MORE2 ASSIGN      4,V27         UPDATE STATS FOR INTERRUPTED JOB            379
239               ASSIGN      45,V28        UPDATE TIME IN INTERVAL                     380
240               ASSIGN      12,K8         INDICATE JOB TO BE RETURNED TO READY        381
241               RELEASE     5             RELEASE THE CPU                             382
242               TRANSFER    ,DISP5                                                    383
            *                                                                           384
            *                **   SECTION VI-A   **                                     385
            *                                                                           386
            *    RE-ORDER CHAINS            (JOBS)                                    387
            *                                                                           388
243         DISP1 ASSIGN      12,K8         DESIGNATE THIS JOB IS FROM READY QUEUE      389
244         DISP5 TEST NE     P7,K0,DISP2   IS THIS ZERO OR JOB?                        390
245               ASSIGN      7,V35         NEW TEMP DISPATCHING PRIORITY               391
246               TEST E      P7,K0,DISP2   IS TEMP DISPATCHING PRIORITY ZERO?          392
247               ASSIGN      7,K1          ASSIGN MINIMUM NON SYSTEM DISPATCHING       393
248         DISP2 ASSIGN      45,K0         RESET TIME IN INTERVAL                      394
249               ASSIGN      46,K0         RESET I/O REQUESTS IN INTERVAL              395
250         DISP3 LINK        *12,P7        RELINK TO READY QUEUE                       396
251         DISP4 ASSIGN      12,K11        DESIGNATE JOB IS FROM I/O WAIT QUEUE        397
252               TRANSFER    ,DISP5                                                    398
            *                                                                           399
            *                                                                           400
            *     PRINT TEST RESULTS                                                    401
            *                                                                           402
253             GENERATE    V65                                                       403
254             PRINT       ,,F,X                                                     404
255             PRINT       ,,U,X                                                     405
256             PRINT       ,,T,X                                                     406
257             PRINT       10,10,CHA,X                                               407
258             UNLINK      10,NNCC,ALL                                               408
259             TERMINATE                                                             409
260         NNCC  LINK      14,FIFO                                                   410
            *                                                                           411
                START       1,NP                                                      412
                END                                                                   413
```

| BIBLIOGRAPHIC DATA SHEET | 1. Report No. UIUCDCS-R72-528 | 2. | 3. Recipient's Accession No. |
|---|---|---|---|

4. Title and Subtitle

A GPSS SIMULATION OF THE 360/75 UNDER HASP AND O.S. 360

5. Report Date: June 1972

6.


7. Author(s): Fred Salz

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois
Urbana, Illinois 61801


8. Performing Organization Rept. No.

10. Project/Task/Work Unit No.

11. Contract/Grant No.: NSF GJ 28289

12. Sponsoring Organization Name and Address
National Science Foundation
Washington, DC

13. Type of Report & Period Covered: Research

14.

15. Supplementary Notes


16. Abstracts

The success or failure of a computing system in today's highly competitive market is often determined by the efficiency of its operating system. Consequently, existing systems are constantly being modified, extended, and, hopefully, improved. The key question pertaining to the implementation of proposed changes is: "Does the proposed change improve the existing operating system?" One appealing method of answering this question is to simulate the operation of the computing system both under the existing operating system and the system with the proposed changes included. The obvious first step in such a study is to build a model to simulate, with accuracy, the existing system. In this paper, such a model is presented for the IBM 360/75 operating under HASP and O.S. A GPSS simulation of the system is presented and some results are given which verify the accuracy of the simulation.



17. Key Words and Document Analysis. 17a. Descriptors

System Modeling, Simulators

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group


18. Availability Statement: Release unlimited

19. Security Class (This Report) UNCLASSIFIED

20. Security Class (This Page) UNCLASSIFIED

21. No. of Pages: 48

22. Price